

# THE POWER OF ARC CONSISTENCY FOR CSPs DEFINED BY PARTIALLY-ORDERED FORBIDDEN PATTERNS

MARTIN C. COOPER<sup>1</sup> AND STANISLAV ŽIVNÝ<sup>2</sup>

<sup>1</sup>IRIT, University of Toulouse III, France  
e-mail address: cooper@irit.fr

<sup>2</sup>Dept. of Computer Science, University of Oxford, UK  
e-mail address: standa.zivny@cs.ox.ac.uk

---

**ABSTRACT.** Characterising tractable fragments of the constraint satisfaction problem (CSP) is an important challenge in theoretical computer science and artificial intelligence. Forbidding patterns (generic sub-instances) provides a means of defining CSP fragments which are neither exclusively language-based nor exclusively structure-based. It is known that the class of binary CSP instances in which the broken-triangle pattern (BTP) does not occur, a class which includes all tree-structured instances, are decided by arc consistency (AC), a ubiquitous reduction operation in constraint solvers. We provide a characterisation of simple partially-ordered forbidden patterns which have this AC-solvability property. It turns out that BTP is just one of five such AC-solvable patterns. The four other patterns allow us to exhibit new tractable classes.

## 1. INTRODUCTION

The *constraint satisfaction problem* (CSP) provides a common framework for many theoretical problems in computer science as well as for many real-life applications. A CSP instance consists of a number of variables, a domain, and constraints imposed on the variables with the goal to determine whether the instance is satisfiable, that is, whether there is an assignment of domain values to all the variables in such a way that all the constraints are satisfied.

The general CSP is NP-complete and thus a major research direction is to identify restrictions on the CSP that render the problem *tractable*, that is, solvable in polynomial time.

A substantial body of work exists from the past two decades on applications of universal algebra in the computational complexity of and the applicability of algorithmic paradigms to CSPs. Moreover, a number of celebrated results have been obtained through this method; see [Barto(2014)] for a recent survey. However, the algebraic approach to CSPs is only applicable to *language-based* CSPs, that is, classes of CSPs defined by the set of allowed

---

*1998 ACM Subject Classification:* Logic and constraint programming.

*Key words and phrases:* arc consistency, constraint satisfaction problem, forbidden pattern, tractability. An extended abstract of this article will appear in Proc. of LICS'16.

<sup>1</sup>supported by EPSRC grant EP/L021226/1.

<sup>2</sup>supported by EPSRC grant EP/L021226/1 and a Royal Society University Research Fellowship. Part of this work was done while the second author was visiting the Simons Institute for the Theory of Computing at UC Berkeley.

constraint relations but with arbitrary interactions of the constraint scopes. For instance, the well-known 2-SAT problem is a class of language-based CSPs on the Boolean domain  $\{0, 1\}$  with all constraint relations being *binary*, that is, of arity at most two.

On the other side of the spectrum are *structure-based* CSPs, that is, classes of CSPs defined by the allowed interactions of the constraint scopes but with arbitrary constraint relations. Here the methods that have been successfully used to establish complete complexity classifications come from graph theory [Grohe(2007), Marx(2013)].

The complexity of CSPs that are neither language-based nor structure-based, and thus are often called *hybrid* CSPs, is much less understood; see [Carbonnel and Cooper(2015)] for a recent survey. One approach to hybrid CSPs that has been rather successful studies the classes of CSPs defined by *forbidden patterns*; that is, by forbidding certain generic subinstances. The focus of this paper is on such CSPs. We remark that we deal with *binary* CSPs but, unlike in most papers on (the algebraic approach to) language-based CSPs, the domain is *not* fixed and is part of the input.

An example of a pattern is given in Figure 1(a). This is the so-called *broken triangle pattern* (BTP) [Cooper et al.(2010b)] (a formal definition is given in Section 2). BTP is an example of a *tractable* pattern, which means that any binary CSP instance in which BTP does not occur is solvable in polynomial time. The class of CSP instances defined by forbidding BTP includes, for instance, all tree-structured binary CSPs [Cooper et al.(2010b)]. There are several generalisations of BTP, for instance, to quantified CSPs [Gao et al.(2011)], to existential patterns [Cohen et al.(2015a)], to patterns on more variables [Cooper et al.(2014)], and other classes [Naanaa(2013), Cooper et al.(2015b)].

The framework of forbidden patterns is general enough to capture language-based CSPs in terms of their polymorphisms. For instance, the pattern in Figure 1(b) captures the notion of binary relations that are max-closed [Jeavons and Cooper(1995)].

Surprisingly, there are essentially only two classes of algorithms (and their combinations) known for establishing tractability of CSPs. These are, firstly, a generalisation of Gaussian elimination [Bulatov and Dalmau(2006), Dalmau(2006)], whose applicability for language-based CSPs is known [Idziak et al.(2010)], and, secondly, problems solvable by *local consistency methods*, which originated in artificial intelligence; see references in [Rossi et al.(2006)]. The latter can be defined in many equivalent ways including pebble games, Datalog, treewidth, and proof complexity [Feder and Vardi(1998)]. Intuitively, a class of CSP instances is solvable by  $k$ -consistency if unsatisfiable instances can always be refuted while only keeping partial solutions of size  $k$  “in memory”. For instance, the 2-SAT problem is solvable by local consistency methods.

For structure-based CSPs, the power of consistency methods is well understood: a class of structures can be solved by  $k$ -consistency if and only if the treewidth (modulo homomorphic equivalence) is at most  $k$  [Atserias et al.(2007)]. Consequently, consistency methods solve all tractable cases of structurally-restricted bounded-arity CSPs [Grohe(2007)]. For language-restricted CSPs, the power of consistency methods has only recently been characterised [Barto and Kozik(2014), Bulatov(2009)].

**Contributions.** Our ultimate goal is to understand the power of local consistency methods for hybrid CSPs. On this quest, we focus in this article on the power of the first level of local consistency, known as *arc consistency* (AC), for classes of binary hybrid CSPs defined by forbidden (partially-ordered) patterns.

The class of CSPs defined by forbidding BTP from Figure 1(a) is in fact solvable by AC. But as it turns out, BTP is not the only pattern with this property.

As our main contribution, we give, in Theorem 5.5, a *complete characterisation* of so-called simple partially-ordered forbidden patterns which have this AC-solvability property. Here the partial orders are on variables and domain values. It turns out that BTP is just one of five such AC-solvable patterns. The four other patterns allow us to exhibit new tractable classes, one of which in particular we expect to lead to new applications since it defines a strict generalisation of binary max-closed constraints which have already found applications in computer vision [Cooper(1999)] and temporal reasoning [Dechter et al.(1991)]. We also provide results on the associated meta problem of deciding whether a CSP instance falls into one of these new tractable classes.

Given that AC is the first level of local consistency methods<sup>1</sup> and is implemented in *all* constraint solvers, an understanding of the power of AC is paramount. We note that focusing on classes of CSPs defined by forbidden patterns is very natural as AC *cannot* introduce forbidden patterns. While simple patterns do not cover all partially-ordered patterns it is a natural, interesting, and broad enough concept that covers BTP and four other novel and non-trivial tractable classes. We expect our results and techniques to be used in future work on the power of AC.

**Related work.** Computational complexity classifications have been obtained for binary CSPs defined by forbidden negative patterns (i.e., only pairwise incompatible assignments are specified) [Cohen et al.(2012)] and for binary CSPs defined by patterns on 2 constraints [Cooper and Escamocher(2015)]. Moreover, (generalisations of) forbidden patterns have been studied in the context of variable and value elimination rules [Cohen et al.(2015a)]. Finally, the idea of forbidding patterns as topological minors has recently been investigated [Cohen et al.(2015b)].

[Kolmogorov et al.(2015), Takhanov(2015)] recently considered the possible extensions of the algebraic approach from the language to the hybrid setting.

The power of the valued version of AC [Cooper et al.(2010a)] has recently been characterised [Kolmogorov et al.(2015b)]. Moreover, the valued version of AC is known to solve all tractable finite-valued language-based CSPs [Thapper and Živný(2016)].

## 2. PRELIMINARIES

**2.1. CSPs and patterns.** A pattern can be seen as a generalisation of the concept of a binary CSP instance that leaves the consistency of some assignments to pairs of variables undefined.

**Definition 1.** A *pattern* is a four-tuple  $\langle X, D, A, \text{cpt} \rangle$  where:

- $X$  is a finite set of *variables*;
- $D$  is a finite set of *values*;
- $A \subseteq X \times D$  is the set of possible variable-value assignments called *points*; the *domain* of  $x \in X$  is its non-empty set  $D(x)$  of possible values:  $D(x) = \{a \in D \mid \langle x, a \rangle \in A\}$ ;

---

<sup>1</sup>In some AI literature AC is the second level, the first being *node consistency* [Rossi et al.(2006)]. AC is also the first level for *relational width* [Bulatov(2006)].

- $\text{cpt}$  is a partial *compatibility function* from the set of unordered pairs of points  $\{\{\langle x, a \rangle, \langle y, b \rangle\} \mid x \neq y\}$  to  $\{\text{TRUE}, \text{FALSE}\}$ . If  $\text{cpt}(\langle x, a \rangle, \langle y, b \rangle) = \text{TRUE}$  (resp.,  $\text{FALSE}$ ) we say that  $\langle x, a \rangle$  and  $\langle y, b \rangle$  are *compatible* (resp., *incompatible*). For simplicity, we write  $\text{cpt}(p, q)$  for  $\text{cpt}(\{p, q\})$ .

We will use a simple figurative drawing for patterns. Each variable will be drawn as an oval containing dots for each of its possible points. Pairs in the domain of the function  $\text{cpt}$  will be represented by lines between points: solid lines (called *positive*) for compatibility and dashed lines (called *negative*) for incompatibility.

**Example 1.** The pattern in Figure 11 is called LX. It consists of three variables, five points, six positive edges, and two negative edges.

We refine patterns to give a definition of a CSP instance.

**Definition 2.** A *binary CSP instance*  $P$  is a pattern  $\langle X, D, A, \text{cpt} \rangle$  where  $\text{cpt}$  is a total function, i.e. the domain of  $\text{cpt}$  is precisely  $\{\{\langle x, a \rangle, \langle y, b \rangle\} \mid x \neq y, a \in D(x), b \in D(y)\}$ .

- The relation  $R_{x,y} \subseteq D(x) \times D(y)$  on  $\langle x, y \rangle$  is  $\{\langle a, b \rangle \mid \text{cpt}(\langle x, a \rangle, \langle y, b \rangle) = \text{TRUE}\}$ .
- A *partial solution* to  $P$  on  $Y \subseteq X$  is a mapping  $s : Y \rightarrow D$  where, for all  $x \neq y \in Y$  we have  $\langle s(x), s(y) \rangle \in R_{x,y}$ .
- A *solution* to  $P$  is a partial solution on  $X$ .

For notational simplicity we have assumed that there is *exactly one* binary constraint between each pair of variables. In particular, this means that the absence of a constraint between variables  $x, y$  is modelled by a complete relation  $R_{x,y} = D(x) \times D(y)$  allowing every possible pair of assignments to  $x$  and  $y$ . We say that there is a *non-trivial* constraint on variables  $x, y$  if  $R_{x,y} \neq D(x) \times D(y)$ . We also use the simpler notation  $R_{ij}$  for  $R_{x_i, x_j}$ .

The main focus of this paper is on ordered patterns, which additionally allow for variable and value orders.

**Definition 3.** An *ordered pattern* is a six-tuple  $\langle X, D, A, \text{cpt}, <_X, <_D \rangle$  where:

- $\langle X, D, A, \text{cpt} \rangle$  is a pattern;
- $<_X$  is a (possibly partial) strict order on  $X$ ; and
- $<_D$  is a (possibly partial) strict order on  $D$ .

A pattern  $\langle X, D, A, \text{cpt} \rangle$  can be seen as an ordered pattern with empty variable and value orders, i.e.  $\langle X, D, A, \text{cpt}, \emptyset, \emptyset \rangle$ .

Throughout the paper when we say “pattern” we implicitly mean “ordered pattern” and use the word “unordered” to emphasize, if needed, that the pattern in question is not ordered.

We do not consider patterns with structure (such as equality or order) between elements in the domains of *distinct* variables.

**Definition 4.** A pattern  $P = \langle X, D, A, \text{cpt}, <_X, <_D \rangle$  is called *basic* if (1)  $D(x)$  and  $D(y)$  do not intersect for distinct  $x, y \in X$ , and (2)  $<_D$  only contains pairs of elements  $\langle a, b \rangle$  from the domain of the same variable, i.e.,  $a, b \in D(x)$  for some  $x \in X$ .

**Example 2.** The pattern shown in Figure 1(a) is known as the *broken triangle pattern* (BTP) [Cooper et al.(2010b)]. BTP consists of three variables, four points, three positive edges, two negative edges,  $<_X = \{x < z, y < z\}$ , and  $<_D = \emptyset$ . Given a basic pattern, we can refer to a point  $\langle x, a \rangle$  in the pattern as simply  $a$  when the variable is clear from the context or a figure. For instance, the point  $\langle z, \gamma \rangle$  in Figure 1(a) can be referred to as  $\gamma$ .

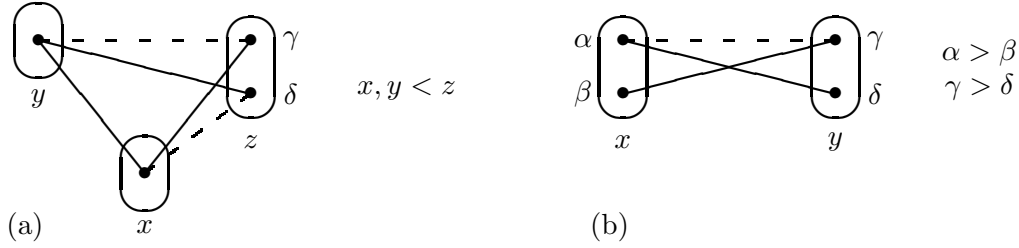


Figure 1: Two AC-solvable patterns: (a) BTP (b) MC.

**Example 3.** The pattern in Figure 1(b) is the (binary) *max-closed* pattern (MC). The pattern MC consists of two variables, four points, two positive edges, one negative edge,  $<_X = \emptyset$ , and  $<_D = \{\beta < \alpha, \delta < \gamma\}$ . MC (Figure 1(b)) together with the extra structure  $\alpha > \gamma$  is an example of a pattern that is not basic.

For some of the proofs we will require patterns with additional structure, namely, the ability to enforce certain points to be distinct.

**Definition 5.** A pattern with a disequality structure is a seven-tuple  $\langle X, D, A, \text{cpt}, <_X, <_D, \neq_D \rangle$  where:

- $\langle X, D, A, \text{cpt}, <_X, <_D \rangle$  is a pattern; and
- $\neq_D \subseteq D \times D$  is a set of pairs of domain values that are distinct.

An example of such a pattern is given in Figure 18(b).

**2.2. Pattern occurrence.** Some points in a pattern are indistinguishable with respect to the rest of the pattern.

**Definition 6.** Two points  $a, b \in D(x)$  are *mergeable* in a pattern  $\langle X, D, A, \text{cpt}, <_X, <_D \rangle$  if there is no point  $p \in A$  for which  $\text{cpt}(\langle x, a \rangle, p)$ ,  $\text{cpt}(\langle x, b \rangle, p)$  are both defined and  $\text{cpt}(\langle x, a \rangle, p) \neq \text{cpt}(\langle x, b \rangle, p)$ .

**Definition 7.** A pattern is called *unmergeable* if it does not contain any mergeable points.

**Example 4.** The points  $\gamma$  and  $\delta$  in BTP (Figure 1(a)) are not mergeable since they have different compatibility with, for instance, the point in variable  $x$ . The pattern LX (Figure 11) is unmergeable.

Some points in a pattern (known as dangling points) are redundant in arc-consistent CSP instances and hence can be removed.

**Definition 8.** Let  $P = \langle X, D, A, \text{cpt}, <_X, <_D \rangle$  be a pattern. A point  $p \in A$  is called *dangling* if it is not ordered by  $<_D$  and if there is at most one point  $q \in A$  for which  $\text{cpt}(p, q)$  is defined, and furthermore (if defined)  $\text{cpt}(p, q) = \text{TRUE}$ .

**Example 5.** The point  $\beta$  in the pattern MC (Figure 1(b)) is not dangling since it is ordered.

In order to use (the absence of) patterns for AC-solvability we need to define what we mean when we say that a pattern *occurs* in a CSP instance. We define the slightly more general notion of occurrence of a pattern in another pattern, thus extending the definitions for unordered patterns [Cooper and Escamocher(2015)]. Recall that a CSP instance corresponds to the special case of a pattern whose compatibility function is total. Essentially

pattern  $P$  occurs in pattern  $Q$  if  $P$  is homomorphic to a subpattern of  $Q$  via an injective renaming of variables and a (possibly non-injective) renaming of points [Cohen et al.(2012)]. We first make the observation that dangling points in a pattern provide no useful information since we assume that all CSP instances are arc consistent, which explains why dangling points can be eliminated from patterns.

**Definition 9.** A pattern is *simple* if it is (i) basic, (ii) has no mergeable points, and (iii) has no dangling points.

From a given pattern it is possible to create an infinite number of equivalent patterns by adding dangling points or by duplicating points. By restricting our attention to simple patterns we avoid having to consider such patterns.

**Definition 10.** Let  $P' = \langle X', D', A', \text{cpt}', <_{X'}, <_{D'} \rangle$  and  $P = \langle X, D, A, \text{cpt}, <_X, <_D \rangle$  be two patterns. A *homomorphism* from  $P'$  to  $P$  is a mapping  $f : A' \rightarrow A$  which satisfies:

- If  $\text{cpt}'(p, q)$  is defined, then  $\text{cpt}(f(p), f(q)) = \text{cpt}'(p, q)$ .
- The mapping  $f_{\text{var}} : X' \rightarrow X$ , given by  $f_{\text{var}}(x') = x$  if  $\exists a', a$  such that  $f(\langle x', a' \rangle) = \langle x, a \rangle$ , is well-defined and injective.
- If  $x' <_{X'} y'$  then  $f_{\text{var}}(x') <_X f_{\text{var}}(y')$ .
- If  $a', b' \in D'(x')$ ,  $a' <_{D'} b'$ ,  $f(\langle x', a' \rangle) = \langle x, a \rangle$  and  $f(\langle x', b' \rangle) = \langle x, b \rangle$  then  $a <_D b$ .

A *consistent linear extension* of a pattern  $P = \langle X, D, A, \text{cpt}, <_X, <_D \rangle$  is a pattern  $P^t$  obtained from  $P$  by first identifying any number of pairs of points  $p, q$  which are both mergeable and incomparable (according to  $<_D$ ) and then extending the orders on the variables and the domain values to total orders.

**Definition 11.** Let  $P' = \langle X', D', A', \text{cpt}', <_{X'}, <_{D'} \rangle$  and  $P = \langle X, D, A, \text{cpt}, <_X, <_D \rangle$  be two patterns.  $P'$  *occurs* in  $P$  if for all consistent linear extensions  $P^t$  of  $P$ , there is a homomorphism from  $P'$  to  $P^t$ . We use the notation  $\text{CSP}_{\overline{\text{SP}}}(P)$  to represent the set of binary CSP instances in which the pattern  $P$  does *not* occur.

This definition extends in a natural way to patterns with a disequality structure.

**Remark 2.1.** We can add  $a \neq b$  to a pattern, without changing its semantics, when  $b <_D a$  or  $\langle x, a \rangle$  and  $\langle x, b \rangle$  are joined by negative and positive edges to some point  $\langle y, c \rangle$ . Furthermore, all domain values  $a, b$  in an *instance* are distinct so there is an implicit  $a \neq b$ .

**Example 6.** The pattern MC (Figure 1(b)) occurs in pattern EMC (Figure 3) but not in patterns BTP (Figure 1(a)) or BTX (Figure 7).

For a pattern  $P$ , we denote by  $\text{unordered}(P)$  the underlying unordered pattern, that is,

$$\text{unordered}(\langle X, D, A, \text{cpt}, <_X, <_D \rangle) = \langle X, D, A, \text{cpt} \rangle.$$

For instance, the pattern  $\text{unordered}(\text{BTP})$  is the pattern from Figure 1(a) *without* the structure  $x, y < z$ .

The following three simple lemmas follow from the definitions.

**Lemma 2.2.** If  $P$  occurs in  $Q$  and  $Q$  occurs in  $R$ , then  $P$  occurs in  $R$ .

**Lemma 2.3.** If  $P$  occurs in  $Q$  and  $P$  does not occur in  $I$ , then  $Q$  does not occur in  $I$ , i.e.  $\text{CSP}_{\overline{\text{SP}}}(P) \subseteq \text{CSP}_{\overline{\text{SP}}}(Q)$ .

**Lemma 2.4.** For any pattern  $P$ ,  $\text{unordered}(P)$  occurs in  $P$ .

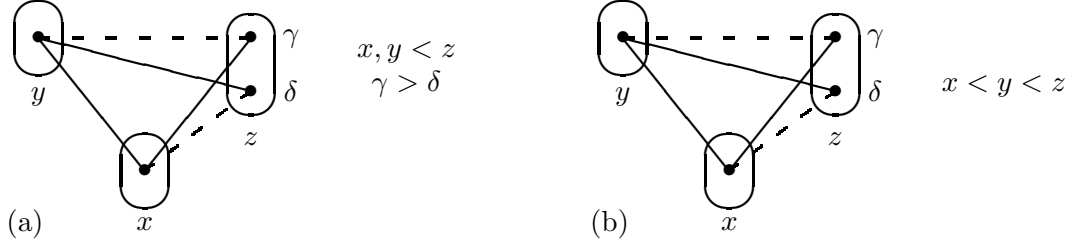


Figure 2: Two equivalent versions of the broken triangle property: forbidding the pattern (a)  $BTP^{do}$  or forbidding the pattern (b)  $BTP^{vo}$  defines the same class of instances.

**2.3. AC solvability.** Arc consistency (AC) is a fundamental concept for CSPs.

**Definition 12.** Let  $I = \langle X, D, A, \text{cpt} \rangle$  be a CSP instance. A point  $\langle x, a \rangle \in A$  is called *arc consistent* if, for all variables  $y \neq x$  in  $X$  there is some point  $\langle y, b \rangle \in A$  compatible with  $\langle x, a \rangle$ .

The CSP instance  $\langle X, D, A, \text{cpt} \rangle$  is called *arc consistent* if  $A \neq \emptyset$  and every point in  $A$  is arc consistent.

Points that are not arc-consistent cannot be part of a solution so can safely be removed. There are optimal  $O(cd^2)$  algorithms for establishing arc consistency which repeatedly remove such points [Bessière et al.(2005)], where  $c$  is the number of non-trivial constraints and  $d$  the maximum domain size. Algorithms establishing arc consistency are implemented in all constraint solvers.

AC is a *decision procedure* for a CSP instance if, after establishing arc consistency, non-empty domains for all variables guarantee the existence of a solution. (Note that a solution can then be found without backtrack by maintaining AC during search). AC is a decision procedure for a class of CSP instances if AC is a decision procedure for every instance from the class.

**Definition 13.** A pattern  $P$  is called *AC-solvable* if AC is a decision procedure for  $\text{CSP}_{\overline{SP}}(P)$ .

The following lemma is a straightforward consequence of the definitions.

**Lemma 2.5.** A pattern  $P$  is not AC-solvable if and only if there is an instance  $I \in \text{CSP}_{\overline{SP}}(P)$  that is arc consistent and has no solution.

The following lemma follows directly from Lemmas 2.3 and 2.5.

**Lemma 2.6.** If  $P$  occurs in  $Q$  and  $P$  is not AC-solvable, then  $Q$  is not AC-solvable.

As our main result we will, in Theorem 5.5, characterise all simple patterns that are AC-solvable.

**2.4. Pattern symmetry and equivalence.** For an ordered pattern  $P$ , we denote by  $\text{invDom}(P)$ ,  $\text{invVar}(P)$  the patterns obtained from  $P$  by inverting the domain order or the variable order, respectively.

**Lemma 2.7.** If  $P$  is not AC-solvable, then neither is any of  $\text{invDom}(P)$ ,  $\text{invVar}(P)$  or  $\text{invDom}(\text{invVar}(P))$ .

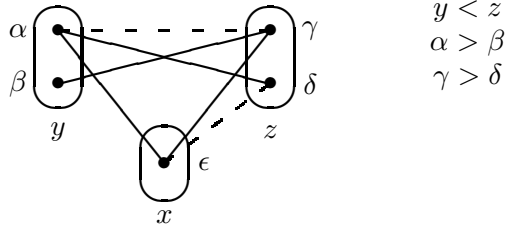


Figure 3: The ordered pattern EMC (Extended Max-Closed)

*Proof.* The claims follow from inverting the respective orders in the instance  $I$  of Lemma 2.5 proving that  $P$  is not AC-solvable.  $\square$

Some patterns define the same classes of CSP instances.

**Definition 14.** Patterns  $P$  and  $P'$  are *equivalent* if

$$\text{CSP}_{\overline{SP}}(P) = \text{CSP}_{\overline{SP}}(P').$$

**Lemma 2.8.** If  $P$  occurs in  $P'$  and  $P'$  occurs in  $P$ , then  $P, P'$  are equivalent.

**Example 7.** Let  $\text{LX}^<$  be the pattern obtained from  $\text{LX}$  (Figure 11) by adding the partial variable order  $y < z$ . Due to the symmetry of  $\text{LX}$ , observe that  $\text{LX}$  and  $\text{LX}^<$  are equivalent.

**Example 8.** The two patterns shown in Figure 2 are also equivalent: (a)  $\text{BTP}^{do}$  with structure  $x, y < z$  and  $c < d$ , and (b)  $\text{BTP}^{vo}$  with variable order  $x < y < z$ . We will call these the *variable-ordered* and *domain-ordered* versions of BTP, respectively, when it is necessary to make the distinction between the two. BTP (Figure 1(a)) will refer to the same pattern with the only structure  $x, y < z$  which again, by symmetry, is equivalent to both  $\text{BTP}^{do}$  and  $\text{BTP}^{vo}$ .

### 3. NEW TRACTABLE CLASSES SOLVED BY ARC CONSISTENCY

Our search for a characterisation of all simple patterns decided by arc consistency surprisingly uncovered four new tractable patterns, which we describe in this section. The first pattern we study is shown in Figure 3. It is a proper generalisation of the MC pattern (Figure 1(b)) since it has an extra variable and three extra edges.

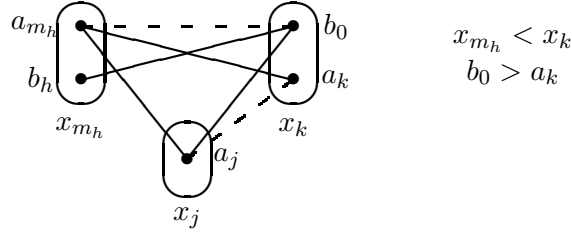
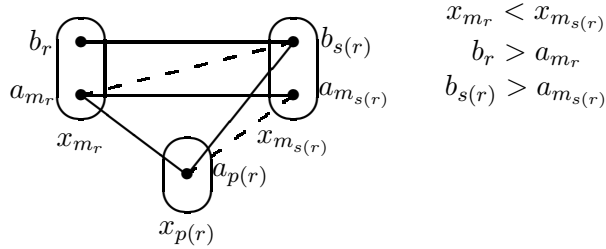
**Theorem 3.1.** AC is a decision procedure for  $\text{CSP}_{\overline{SP}}(\text{EMC})$  where EMC is the pattern shown in Figure 3.

*Proof.* Since establishing arc consistency only eliminates domain elements, and hence cannot introduce the pattern, it suffices to show that every arc-consistent instance  $I = \langle X, D, A, \text{cpt} \rangle \in \text{CSP}_{\overline{SP}}(\text{EMC})$  has a solution. We give a constructive proof. Let  $x_1 < \dots < x_n$  be an ordering of  $X$  such that EMC does not occur in  $I$ . Define an assignment  $\langle a_1, \dots, a_n \rangle$  to the variables  $\langle x_1, \dots, x_n \rangle$  recursively as follows:  $a_1 = \max(D(x_1))$  and, for  $i > 1$ ,

$$a_i = \min\{a_i^j \mid 1 \leq j < i\}, \quad \text{where} \quad a_i^j = \max\{a \in D(x_i) \mid (a_j, a) \in R_{ji}\} \quad (3.1)$$

For  $i > 1$ , we denote by  $\text{pred}(i)$  a value of  $j < i$  such that  $a_i = a_i^j$ . Arc consistency guarantees that  $a_i^j$  exists and hence that  $a_i$  and  $\text{pred}(i)$  are well defined. We claim that



Figure 4: To avoid the pattern EMC, we must have  $b_h > a_{m_h}$ .Figure 5: The situation corresponding to hypothesis  $H_r$ .

$\langle a_1, \dots, a_n \rangle$  is a solution. Suppose, for a contradiction, that  $(a_j, a_k) \notin R_{jk}$  for some  $1 \leq j < k \leq n$ . If there is more than one such pair  $(j, k)$ , then choose  $k$  to be minimal and then for this value of  $k$  choose  $j$  to be minimal.

We prove our claim that  $\langle a_1, \dots, a_n \rangle$  is a solution to  $I$  by induction on  $n$ . The claim trivially holds for  $n = 1$  since  $a_1 \in D(x_1)$ . It remains to show that if the claim holds for instances of size less than  $n$  then it holds for instances of size  $n$ .

Let  $m_0 = k$  and  $m_r = \text{pred}(m_{r-1})$  for  $r \geq 1$  if  $m_{r-1} > 1$ . Let  $t$  be such that  $m_t = 1$ . By definition of  $\text{pred}$ , we have

$$1 = m_t < m_{t-1} < \dots < m_1 < m_0 = k$$

which implies that this series is finite and hence that  $t$  is well-defined.

We distinguish two cases: (1)  $j > m_1$ , and (2)  $j < m_1$ . Since  $(a_j, a_k) \notin R_{jk}$  and  $(a_{m_1}, a_k) \in R_{jk}$  we know that  $j \neq m_1$ .

Case (1)  $j > m_1$ : Define  $b_0 = a_k^j$ . By definition of  $a_k$ , we know that  $a_k \leq a_k^j$ . Since  $(a_j, a_k) \notin R_{jk}$  and  $(a_j, a_k^j) \in R_{jk}$ , we have  $b_0 = a_k^j > a_k$ .

By our choice of  $j$  to be minimal, and since  $j > m_1$  we know that  $(a_{m_r}, a_k) \in R_{m_r k}$  for  $r = 1, \dots, t$ . Indeed, by minimality of  $k$ , we already had  $(a_{m_r}, a_{m_s}) \in R_{m_r m_s}$  for  $1 \leq s \leq r \leq t$ . Thus, since  $k = m_0$ , we have

$$(a_{m_r}, a_{m_s}) \in R_{m_r m_s} \quad \text{for } 0 \leq s \leq r \leq t. \quad (3.2)$$

By arc consistency,  $\exists b_1 \in D(x_{m_1})$  such that  $(b_1, b_0) \in R_{m_1 k}$ . We have  $(a_{m_1}, a_j) \in R_{m_1 j}$  by minimality of  $k$  and since  $m_1, j < k$ . Since  $m_1 = \text{pred}(k)$  and hence  $a_k = a_k^{m_1}$ , we have  $(a_{m_1}, a_k) \in R_{m_1 k}$  and  $(a_{m_1}, b_0) \notin R_{m_1 k}$  by the maximality of  $a_k^{m_1}$  in Equation (3.1). We thus have the situation illustrated in Figure 4 for  $h = 1$ . Since the pattern EMC does not occur in  $I$ , we must have  $b_1 > a_{m_1}$ .

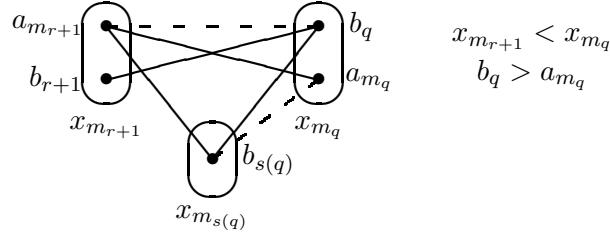


Figure 6: To avoid the pattern EMC, we must have  $b_{r+1} > a_{m_{r+1}}$ .

For  $1 \leq r \leq t$ , let  $H_r$  be the following hypothesis.

$H_r$ :  $\exists s(r) \in \{0, \dots, r-1\}$ ,  $\exists p(r) < k$ ,  $\exists b_r \in D(x_{m_r})$ , with  $b_r > a_{m_r}$ , such that we have the situation shown in Figure 5.

We have just shown that  $H_1$  holds (with  $s(1) = 0$  and  $p(1) = j$ ). We now show, for  $1 \leq r < t$ , that  $(H_1 \wedge \dots \wedge H_r) \Rightarrow H_{r+1}$ .

We know that  $(a_{m_{r+1}}, a_{m_r}) \in R_{m_{r+1}m_r}$  and  $(a_{m_{r+1}}, b_r) \notin R_{m_{r+1}m_r}$ , since  $m_{r+1} = \text{pred}(m_r)$  and by maximality of  $a_{m_r} = a_{m_r}^{m_{r+1}}$  in Equation (3.1). Let  $q \in \{0, \dots, r\}$  be minimal such that  $(a_{m_{r+1}}, b_q) \notin R_{m_{r+1}m_q}$ . We distinguish two cases: (a)  $q = 0$ , and (b)  $q > 0$ .

If  $q = 0$ , then we have  $(a_{m_{r+1}}, a_k) \in R_{m_{r+1}k}$  (from Equation (3.2), since  $k = m_0$ ),  $(a_{m_{r+1}}, b_0) \notin R_{m_{r+1}k}$  (since  $q = 0$ ),  $(a_{m_{r+1}}, a_j) \in R_{m_{r+1}j}$  (by minimality of  $k$ , since  $m_{r+1}, j < k$ ). By arc consistency,  $\exists b_{r+1} \in D(x_{m_{r+1}})$  such that  $(b_{r+1}, b_0) \in R_{m_{r+1}k}$ . We then have the situation illustrated in Figure 4 for  $h = r+1$ . As above, from the absence of pattern EMC, we can deduce that  $b_{r+1} > a_{m_{r+1}}$ . We thus have  $H_{r+1}$  (with  $s(r+1) = 0$  and  $p(r+1) = j$ ).

If  $q > 0$ , then  $H_1 \wedge \dots \wedge H_r$  implies that  $H_q$  holds. By minimality of  $q$ , we know that  $(a_{m_{r+1}}, b_{s(q)}) \in R_{m_{r+1}m_{s(q)}}$  since  $s(q) < q$ . We know that  $(a_{m_{r+1}}, a_{m_q}) \in R_{m_{r+1}m_q}$  from Equation (3.2), and that  $(a_{m_{r+1}}, b_q) \notin R_{m_{r+1}m_q}$  by definition of  $q$ . We know that  $(b_q, b_{s(q)}) \in R_{m_qm_{s(q)}}$  and  $(a_{m_q}, b_{s(q)}) \notin R_{m_qm_{s(q)}}$  from  $H_q$ . By arc consistency,  $\exists b_{r+1} \in D(x_{m_{r+1}})$  such that  $(b_{r+1}, b_q) \in R_{m_{r+1}m_q}$ . We then have the situation illustrated in Figure 6. As above, from the absence of pattern EMC, we can deduce that  $b_{r+1} > a_{m_{r+1}}$ . We thus have  $H_{r+1}$  (with  $s(r+1) = q$  and  $p(r+1) = s(q)$ ).

Case (2)  $j < m_1$ : Consider the subproblem  $I'$  of  $I$  on variables  $\{x_1, x_2, \dots, x_{m_1-1}\} \cup \{x_k\}$ . Since  $x_{m_1}$  does not belong to the set of variables of  $I'$ , this instance has size strictly less than  $n$ , and hence by our inductive hypothesis has a solution. The values of  $a_i$  may differ between  $I$  and  $I'$ . However, we can see from its definition given in Equation (3.1), that the value of  $a_i$  depends uniquely on the subproblem on previous variables  $\{x_1, \dots, x_{i-1}\}$ . Showing the dependence on the instance by a superscript, we thus have  $a_i^{I'} = a_i^I$  ( $i = 1, \dots, m_1 - 1$ ) although  $a_k^{I'}$  may (and, in fact, does) differ from  $a_k^I$ . By our inductive hypothesis,  $\langle a_1, \dots, a_{m_1-1}, a_k^{I'} \rangle$  is a solution to  $I'$ . Setting  $b_0 = a_k^{I'}$ , it follows that  $(a_i, b_0) \in R_{ik}$  for  $1 \leq i < m_1$ . In particular, since  $j < m_1$ , we have  $(a_j, b_0) \in R_{jk}$ . Now  $a_k^I \leq a_k^{I'} = b_0$ , since  $I'$  is a subinstance of  $I$  (and so, from Equation (3.1),  $a_k^I$  is the minimum of a superset over which  $a_k^{I'}$  is a minimum). Thus  $a_k = a_k^I < b_0$ , since  $(a_j, b_0) \in R_{jk}$  and  $(a_j, a_k) \notin R_{jk}$ .

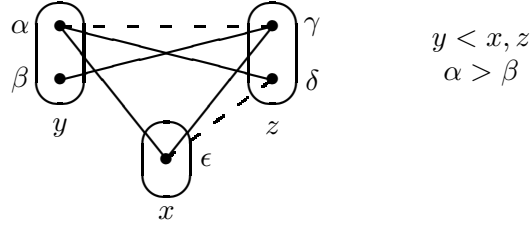


Figure 7: The ordered pattern BTX.

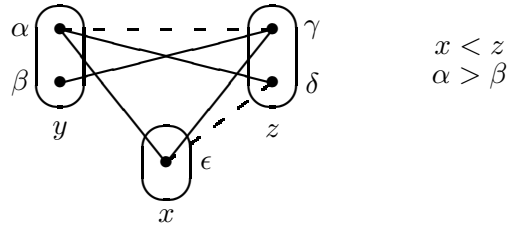


Figure 8: The ordered pattern BTI.

By arc consistency,  $\exists b_1 \in D(x_{m_1})$  such that  $(b_1, b_0) \in R_{m_1 k}$ . As in case (1), we have the situation illustrated in Figure 4 for  $h = 1$ . Since the pattern EMC does not occur in  $I$ , we must have  $b_1 > a_{m_1}$ .

Consider the hypothesis  $H_r$  stated in case (1) and illustrated in Figure 5. We have just shown that  $H_1$  holds (with  $s(1) = 0$  and  $p(1) = j$ ). We now show, for  $1 \leq r < t$ , that  $(H_1 \wedge \dots \wedge H_r) \Rightarrow H_{r+1}$ .

As in case (1), we know that  $(a_{m_{r+1}}, a_{m_r}) \in R_{m_{r+1} m_r}$  and  $(a_{m_{r+1}}, b_r) \notin R_{m_{r+1} m_r}$ . Let  $q \in \{0, \dots, r\}$  be minimal such that  $(a_{m_{r+1}}, b_q) \notin R_{m_{r+1} m_q}$ . We have seen above that  $(a_{m_{r+1}}, b_0) \in R_{m_{r+1} k}$  (since  $x_{m_{r+1}}, x_{m_0}$  are assigned, respectively, the values  $a_{m_{r+1}}, b_0$  in a solution to  $I'$ ). Therefore, we can deduce that  $q > 0$ . Therefore  $H_1 \wedge \dots \wedge H_r$  implies that  $H_q$  holds. By minimality of  $k$ , and since  $m_q < m_0 = k$ , we know that  $(a_{m_{r+1}}, a_{m_q}) \in R_{m_{r+1} m_q}$ . As in case (1), by minimality of  $q$ , we know that  $(a_{m_{r+1}}, b_{s(q)}) \in R_{m_{r+1} m_{s(q)}}$ . By arc consistency,  $\exists b_{r+1} \in D(x_{m_{r+1}})$  such that  $(b_{r+1}, b_q) \in R_{m_{r+1} m_q}$ . We thus have the situation illustrated in Figure 6. Again, from the absence of pattern EMC, we can deduce that  $b_{r+1} > a_{m_{r+1}}$ . We thus again have  $H_{r+1}$  with  $s(r+1) = q$  and  $p(r+1) = s(q)$ .

Thus, by induction on  $r$ , we have shown in both cases that  $H_t$  holds. But recall that  $m_t = 1$  and that  $a_1$  was chosen to be the maximal element of  $D(x_1)$  and hence  $\nexists b_t \in D(x_1)$  such that  $b_t > a_1$ . This contradiction shows that  $\langle a_1, \dots, a_n \rangle$  is a solution, as claimed.  $\square$

The next two patterns we study in this section, shown in Figure 7 and Figure 8, are similar to EMC but the three patterns are incomparable (in the sense that none occurs in another) due to the different orders on the three variables.

**Theorem 3.2.** *AC is a decision procedure for  $CSP_{\overline{SP}}(BTX)$  where  $BTX$  is the pattern shown in Figure 7.*

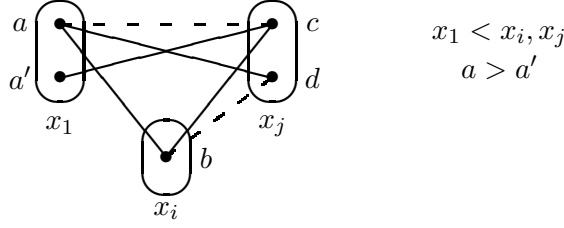


Figure 9: An occurrence of the pattern BTX.

*Proof.* Since establishing arc consistency only eliminates domain elements, and hence cannot introduce the pattern, we only need to show that every arc-consistent instance  $I = \langle X, D, A, \text{cpt} \rangle \in \text{CSP}_{\overline{\text{SP}}}(\text{BTX})$  has a solution. Let  $x_1 < \dots < x_n$  be an ordering of  $X$  such that BTX does not occur in  $I$ . In fact we will show a stronger result by proving that the hypothesis  $H_n$ , below, holds for all  $n \geq 1$ .

$H_n$ : for all arc-consistent instances  $I = \langle X, D, A, \text{cpt} \rangle$  from  $\text{CSP}_{\overline{\text{SP}}}(\text{BTX})$  with  $|X| = n$ , if  $a = \max(D(x_1))$ , then  $I$  has a solution  $s$  with  $s(x_1) = a$ .

Trivially,  $H_1$  holds. Suppose that  $H_{n-1}$  holds where  $n > 1$ . We will show that this implies  $H_n$ , which will complete the proof by induction. Let  $D'(x_i) = \{b \in D(x_i) \mid (a, b) \in R_{1i}\}$  ( $i = 2, \dots, n$ ). Denote by  $I'$  the subproblem of  $I$  on variables  $2, \dots, n-1$  and domains  $D'(x_i)$  ( $i = 2, \dots, n$ ). To complete the inductive proof, it is sufficient to show that  $I'$  is arc consistent: since  $I' \in \text{CSP}_{\overline{\text{SP}}}(\text{BTX})$  and has  $n-1$  variables, by  $H_{n-1}$ , if  $I'$  is arc consistent it has a solution which can clearly be extended to a solution to  $I$  by adding the assignment  $(x_1, a)$ .

Consider any two variables  $x_i, x_j$  such that  $i, j > 1$ . We know by arc consistency of  $I$  that  $D'(x_i)$  is non-empty. Let  $b \in D'(x_i)$  (i.e.  $(a, b) \in R_{1i}$ ). To complete the proof, it suffices to show that  $b$  has a support in  $D'(x_j)$ . By arc consistency of  $I$ , we can deduce the existence of  $c \in D(x_j)$  such that  $(b, c) \in R_{ij}$ , and then  $a' \in D(x_1)$  such that  $(a', c) \in R_{1j}$ , as well as  $d \in D(x_j)$  such that  $(a, d) \in R_{1j}$  (i.e.  $d \in D'(x_j)$ ). If  $b$  has no support in  $D'(x_j)$ , then we must have  $c \notin D'(x_j)$  (i.e.  $(a, c) \notin R_{1j}$ ) and  $(b, d) \notin R_{ij}$ . Since  $a$  is the maximum element of  $D(x_1)$ , we know that  $a' \leq a$ . Indeed, since  $(a, c) \notin R_{1j}$  and  $(a', c) \in R_{1j}$ , we have that  $a' < a$ . But then the pattern BTX occurs in  $I$ , as shown in Figure 9. This contradiction shows that  $b$  does have a support in  $D'(x_j)$  and hence that  $I'$  is arc consistent, as required.  $\square$

**Theorem 3.3.** *AC is a decision procedure for  $\text{CSP}_{\overline{\text{SP}}}(\text{BTI})$  where BTI is the pattern shown in Figure 8.*

*Proof.* Since establishing arc consistency only eliminates domain elements, and hence cannot introduce the pattern, we only need to show that every arc-consistent instance  $I = \langle X, D, A, \text{cpt} \rangle \in \text{CSP}_{\overline{\text{SP}}}(\text{BTI})$  has a solution. Let  $x_1 < \dots < x_n$  be an ordering of  $X$  such that BTI does not occur in  $I$ . In fact, we will show a stronger result by proving that, for all arc-consistent instances  $I = \langle X, D, A, \text{cpt} \rangle \in \text{CSP}_{\overline{\text{SP}}}(\text{BTI})$ ,  $I$  has a solution  $s$  defined recursively by:  $s(x_i)$  is the maximum value in  $D(x_i)$  compatible with all previous assignments  $s(x_1), \dots, s(x_{i-1})$ .

Let  $T_i$  denote the set of values in  $D(x_i)$  compatible with the assignments  $s(x_1), \dots, s(x_{i-1})$  (defined recursively as above). If each  $T_i$  ( $i = 1, \dots, n$ ) is non-empty, then  $s(x_1), \dots, s(x_n)$

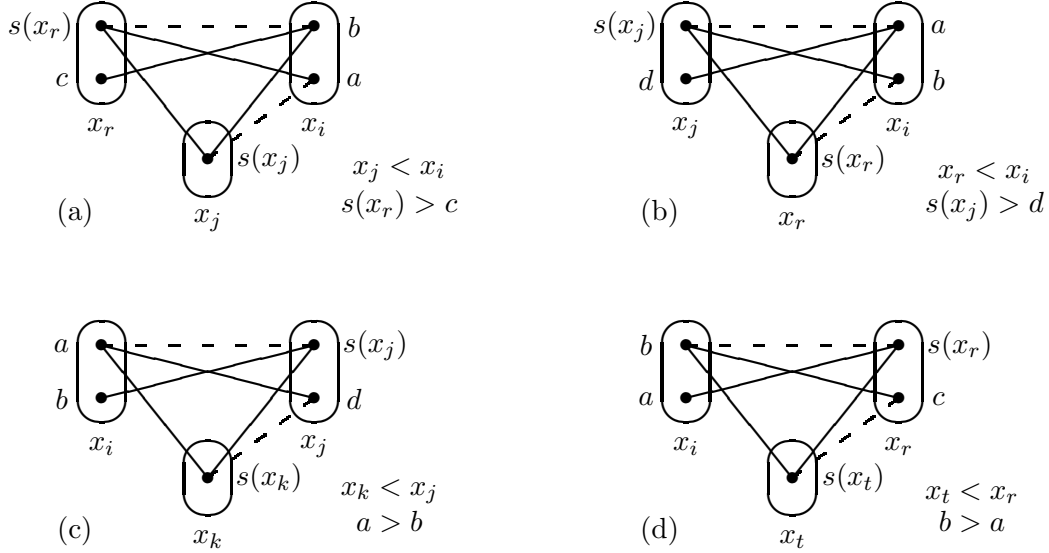


Figure 10: Occurrences of the pattern BTI.

is a solution to  $I$ . Let  $I_i$  denote the subinstance of  $I$  on the first  $i$  variables  $x_1, \dots, x_i$ .  $T_1 = D(x_1)$  and is non-empty since  $I$  is arc consistent, and hence  $s(x_1) = \max(T_1)$  is a solution to  $I_1$ . Suppose that  $s(x_1), \dots, s(x_{i-1})$ , as defined above, is a solution to  $I_{i-1}$ . We will show that  $s(x_i)$  exists and hence that  $s(x_1), \dots, s(x_i)$  is a solution to  $I_i$ , which by a simple induction will complete the proof. Suppose for a contradiction that  $s(x_i)$  does not exist, i.e. that  $T_i = \emptyset$ .

Let  $R_{ji}(u)$  denote the subset of  $D(x_i)$  which is compatible with the assignment of  $u$  to  $x_j$ , i.e.  $R_{ji}(u) = \{v \in D(x_i) \mid (u, v) \in R_{ji}\}$ . For  $j \in \{1, \dots, i-1\}$ , let  $T_i^j$  be the intersection of the sets  $R_{hi}(s(x_h))$  ( $h = 1, \dots, j$ ). By our hypothesis that  $T_i = \emptyset$ , we know that  $T_i^{i-1} = T_i = \emptyset$ . By arc consistency,  $T_i^1 \neq \emptyset$ . Let  $j \in \{2, \dots, i-1\}$  be minimal such that  $T_i^j = \emptyset$ . Thus

$$R_{ji}(s(x_j)) \cap T_i^{j-1} = \emptyset \quad (3.3)$$

By arc consistency,  $\exists b \in D(x_i)$  such that  $(s(x_j), b) \in R_{ji}$  (i.e.  $b \in R_{ji}(s(x_j))$ ). By Equation (3.3) and definition of  $T_i^{j-1}$ , there is some  $r < j$  such that  $(s(x_r), b) \notin R_{ri}$ . Choose  $r$  to be minimal. Consider  $a \in T_i^{j-1}$  (which is non-empty since  $j$  was chosen to be minimal). Then  $(s(x_r), a) \in R_{ri}$ , by definition of  $T_i^{j-1}$ , since  $r \leq j-1$ . But  $(s(x_j), a) \notin R_{ji}$  by (3.3). By arc consistency,  $\exists c \in D_r$  such that  $(c, b) \in R_{ri}$  and  $\exists d \in D(x_j)$  such that  $(d, a) \in R_{ji}$ . By our inductive hypothesis that  $s(x_1), \dots, s(x_{i-1})$  is a solution to  $I_{i-1}$ , we know that  $(s(x_r), s(x_j)) \in R_{rj}$  since  $r, j < i$ . Because of their different compatibilities with, respectively,  $b$  and  $a$ , we know that  $c \neq s(x_r)$  and  $d \neq s(x_j)$ . If  $c < s(x_r)$ , then, since  $r < i$ , the pattern BTI occurs in  $I$ , as shown in Figure 10(a); so we can deduce that  $c > s(x_r)$ . Similarly, if  $d < s(x_j)$ , then, since  $j < i$ , the pattern BTI occurs in  $I$ , as shown in Figure 10(b); so we can deduce that  $d > s(x_j)$ .

But, by definition of  $s$ ,  $s(x_r)$  is the maximal element of  $T_r$ . So, since  $c > s(x_r)$ , there must be some  $t < r$  such that  $(s(x_t), c) \notin R_{tr}$ . By minimality of  $r$ , we know that  $(s(x_t), b) \in R_{ti}$ . Similarly, there must be some  $k < j$  such that  $(s(x_k), d) \notin R_{kj}$  since

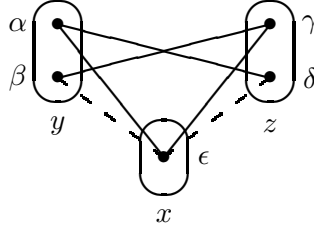


Figure 11: The pattern LX.

$d > s(x_j)$  and  $s(x_j)$  is the maximal element of  $T_j$ . Since  $a \in T_i^{j-1}$  and  $k < j$ , we know that  $(s(x_k), a) \in R_{ki}$ . By our inductive hypothesis that  $s(x_1), \dots, s(x_{i-1})$  is a solution to  $I_{i-1}$ , we know that  $(s(x_k), s(x_j)) \in R_{kj}$  and  $(s(x_t), s(x_r)) \in R_{tr}$  since  $k, j, t, r < i$ . We know that  $a \neq b$  because they have different compatibilities with  $s(x_j)$ . If  $a > b$ , then, since  $k < j$ , the pattern BTI occurs in  $I$ , as shown in Figure 10(c). And, if  $b > a$ , then the pattern BTI occurs in  $I$ , as shown in Figure 10(d). This contradiction shows that  $T_i \neq \emptyset$  (for each  $i = 1, \dots, n$ ) and hence that  $I$  has a solution  $s$ .  $\square$

We conclude this section with a pattern which is essentially different from the patterns EMC, BTX, and BTI, since it includes two negative edges that meet but has no domain or variable order. The tractability of this pattern was previously unknown [Escamocher(2014)].

**Theorem 3.4.** *AC is a decision procedure for  $\text{CSP}_{\overline{\text{SP}}}(LX)$  where  $LX$  is the pattern shown in Figure 11.*

*Proof.* Since establishing arc consistency only eliminates domain elements, and hence cannot introduce the pattern, we only need to show that every arc-consistent instance  $I \in \text{CSP}_{\overline{\text{SP}}}(LX)$  has a solution. In fact we will show a stronger result by proving that the hypothesis  $H_n$ , below, holds for all  $n \geq 1$ .

$H_n$ : for all arc-consistent instances  $I = \langle X, D, A, \text{cpt} \rangle \in \text{CSP}_{\overline{\text{SP}}}(LX)$  with  $|X| = n$ ,  $\forall x_i \in X, \forall a \in D(x_i), I$  has a solution  $s$  such that  $s(x_i) = a$ .

Trivially,  $H_1$  holds. Suppose that  $H_{n-1}$  holds where  $n > 1$ . We will show that this implies  $H_n$ , which will complete the proof by induction.

Consider an arc-consistent instance  $I = \langle X, D, A, \text{cpt} \rangle$  from  $\text{CSP}_{\overline{\text{SP}}}(LX)$  with  $X = \{x_1, \dots, x_n\}$  and let  $a \in D(x_i)$  where  $1 \leq i \leq n$ . Let  $I_{n-1}$  denote the subproblem of  $I$  on variables  $X \setminus \{x_i\}$ . For any solution  $s$  of  $I_{n-1}$ , we denote by  $CV(\langle x_i, a \rangle, s)$  the set of variables in  $X \setminus \{x_i\}$  on which  $s$  is compatible with the unary assignment  $\langle x_i, a \rangle$ , i.e.

$$CV(\langle x_i, a \rangle, s) = \{x_j \in X \setminus \{x_i\} \mid (a, s(x_j)) \in R_{ij}\}$$

Consider two distinct solutions  $s, s'$  to  $I_{n-1}$ . If we have  $x_j \in CV(\langle x_i, a \rangle, s) \setminus CV(\langle x_i, a \rangle, s')$  and  $x_k \in CV(\langle x_i, a \rangle, s') \setminus CV(\langle x_i, a \rangle, s)$ , then the pattern LX occurs in  $I$  under the mapping  $x \mapsto x_i, y \mapsto x_j, z \mapsto x_k, \alpha \mapsto s(x_j), \beta \mapsto s'(x_j), \gamma \mapsto s'(x_k), \delta \mapsto s(x_k), \epsilon \mapsto a$  (see Figure 11). Since LX does not occur in  $I$ , we can deduce that the sets  $CV(\langle x_i, a \rangle, s)$ , as  $s$  varies over all solutions to  $I_{n-1}$ , form a nested family of sets. Let  $s_a$  be a solution to  $I_{n-1}$  such that  $CV(\langle x_i, a \rangle, s_a)$  is maximal for inclusion.

Consider any  $x_j \in X \setminus \{x_i\}$ . By arc consistency,  $\exists b \in D(x_j)$  such that  $(a, b) \in R_{ij}$ . By our inductive hypothesis  $H_{n-1}$ , there is a solution  $s$  to  $I_{n-1}$  such that  $s(x_j) = b$ . Since  $(a, s(x_j)) = (a, b) \in R_{ij}$ , we have  $x_j \in CV(\langle x_i, a \rangle, s)$ . By maximality of  $s_a$ , this implies

$x_j \in CV(\langle x_i, a \rangle, s_a)$ , i.e.  $(a, s_a(x_j)) \in R_{ij}$ . Since this is true for any  $x_j \in X \setminus \{x_i\}$ , we can deduce that  $s_a$  can be extended to a solution to  $I$  (which assigns  $a$  to  $x_i$ ) by simply adding the assignment  $\langle x_i, a \rangle$  to  $s_a$ .  $\square$

#### 4. RECOGNITION PROBLEM FOR UNKNOWN ORDERS

For an unordered pattern  $P$  of size  $k$ , checking for (the non-occurrence of)  $P$  in a CSP instance  $I$  is solvable in time  $O(|I|^k)$  by simple exhaustive search. Consequently, checking for (the non-occurrence of) unordered patterns of constant size is solvable in polynomial time. However, the situation is less obvious for ordered patterns since we have to test all possible orderings of  $I$ .

The following result was shown in [Cooper et al.(2010b)].

**Theorem 4.1.** *Given a binary CSP instance  $I$  with a fixed total order on the domain, there is a polynomial-time algorithm to find a total variable ordering such that BTP does not occur in  $I$  (or to determine that no such ordering exists).*

We show that the same result holds for the other three ordered patterns studied in this paper, namely BTI, BTX, and EMC.

**Theorem 4.2.** *Given a binary CSP instance  $I$  with a fixed total order on the domain and a pattern  $P \in \{BTI, BTX, EMC\}$ , there is a polynomial-time algorithm to find a total variable ordering such that  $P$  does not occur in  $I$  (or to determine that no such ordering exists).*

*Proof.* We give a proof only for BTX as the same idea works for the other two patterns as well. Given a binary CSP instance  $I$  with  $n$  variables  $x_1, \dots, x_n$ , we define an associated CSP instance  $\Pi_I$  that has a solution precisely when there exists a suitable variable ordering for  $I$ . To construct  $\Pi_I$ , let  $O_1, \dots, O_n$  be variables taking values in  $\{1, \dots, n\}$  representing positions in the ordering. We impose the ternary constraint  $O_i > \max(O_j, O_k)$  for all triples of variables  $x_i, x_j, x_k$  in  $I$  such that the BTX pattern occurs for some  $\alpha, \beta \in D(x_i)$  with  $\alpha > \beta$ ,  $\epsilon \in D(x_j)$ , and  $\gamma, \delta \in D(x_k)$  when the variables are ordered  $x_i < x_j, x_k$ . The instance  $\Pi_I$  has a solution precisely if there is an ordering of the variables  $x_1, \dots, x_n$  of  $I$  for which BTX does not occur. Note that if the solution obtained represents a partial order (i.e. if  $O_i$  and  $O_j$  are assigned the same value for some  $i \neq j$ ), then it can be extended to a total order which still satisfies all the constraints by arbitrarily choosing the order of those  $O_i$ 's that are assigned the same value. This reduction is polynomial in the size of  $I$ . We now show that all constraints in  $\Pi_I$  are ternary max-closed and thus  $\Pi_I$  can be solved in polynomial time [Jeavons and Cooper(1995)]. Let  $\langle p_1, q_1, r_1 \rangle$  and  $\langle p_2, q_2, r_2 \rangle$  satisfy any constraint in  $\Pi_I$ . Then  $p_1 > \max(q_1, r_1)$  and  $p_2 > \max(q_2, r_2)$ , and thus  $\max(p_1, p_2) > \max(\max(q_1, r_1), \max(q_2, r_2)) = \max(\max(q_1, q_2), \max(r_1, r_2))$ . Consequently,  $\langle \max(p_1, p_2), \max(q_1, q_2), \max(r_1, r_2) \rangle$  also satisfies the constraint. We can deduce that all constraints in  $\Pi_I$  are max-closed.  $\square$

Using the same technique, we can also show the following.

**Theorem 4.3.** *Given a binary CSP instance  $I$  with a fixed total variable order and a pattern  $P \in \{BTI, BTX\}$ , there is a polynomial-time algorithm to find a total domain ordering such that  $P$  does not occur in  $I$  (or determine that no such ordering exists).*

It is known that determining a domain order for which MC does not occur is NP-hard [Green and Cohen(2008)]. Not surprisingly, for EMC when the domain order is not known, detection becomes NP-hard. For the case of BTX and BTI, if neither the domain nor variable order is known, finding orders for which the pattern does not occur is again NP-hard.

**Theorem 4.4.** *For the pattern EMC, even for a fixed total variable order of an arc-consistent binary CSP instance  $I$ , it is NP-hard to find a total domain ordering of  $I$  such that the pattern does not occur in  $I$ . For patterns BTX and BTI, it is NP-hard to find total variable and domain orderings of an arc-consistent binary CSP instance  $I$  such that the pattern does not occur in  $I$ .*

*Proof.* To show this, we exhibit a polynomial reduction from 3SAT. Given an  $n$ -variable instance  $I_{3SAT}$  of 3SAT, we create a domain of size  $2n$  with a value  $a_i$  for each variable  $X_i$  in  $I_{3SAT}$  and another  $a_{i+n}$  for its negation  $\overline{X_i}$ . This is the domain of each variable  $x_i$  in a binary CSP instance  $I_{CSP}$ . For each total ordering  $>$  of the domain there is a corresponding assignment to the variables of  $I_{3SAT}$  given by  $X_i = \text{true}$  if and only if  $a_i > a_{i+n}$ . To complete the reduction we have to show how to impose a clause, e.g.  $(a_i > a_{i+n}) \vee (a_j > a_{j+n}) \vee (a_k > a_{k+n})$ . The basic construction of  $I_{CSP}$  is composed of the following elements:  $N$  variables linked by equality constraints  $x_i = x_{i+1}$  ( $i = 1, \dots, N-1$ ).

Consider first the case of EMC. To impose a clause we can construct a gadget on any three variables  $x_p, x_q, x_r$  which are not linked by any other constraints in our construction (in particular, not consecutive variables linked by equality constraints). We add four extra values  $a_{\max}, b, c, d$  (of which  $b, c, d$  depend on the clause) in the domain of each of these three variables  $x_p, x_q, x_r$ . The value  $a_{\max}$  is compatible with all values in the domains of the other variables. This ensures arc consistency for all domain values, and if  $a_{\max} > a$  for all domain values  $a \neq a_{\max}$ , then the pattern cannot occur on  $a_{\max}$ . We then place negative edges between all pairs of values other than  $a_{\max}$  except for three positive edges in each constraint. In the constraint between  $x_p$  and  $x_q$  we add the three positive edges:  $(a_i, b), (a_i, c), (a_{i+n}, b)$ . If  $a_{i+n} > a_i$  in  $D(x_p)$  and  $c > b$  in  $D(x_q)$ , then the pattern EMC occurs in  $I_{CSP}$ . The third variable of the pattern is  $x_{q+1}$  or  $x_{q-1}$  which is linked by an equality constraint to  $x_q$ . Thus, to avoid the pattern occurring in  $I_{CSP}$  the domain order must respect

$$(a_i > a_{i+n}) \vee (b > c)$$

By adding similar constraints between  $x_p, x_r$  and  $x_q, x_r$ , we can also impose

$$\begin{aligned} (a_j > a_{j+n}) \vee (c > d) \quad \text{and} \\ (a_k > a_{k+n}) \vee (d > b) \end{aligned}$$

By imposing these three inequalities, we impose  $(a_i > a_{i+n}) \vee (a_j > a_{j+n}) \vee (a_k > a_{k+n})$  (since we cannot simultaneously have  $b > c > d > b$ ) which corresponds to the clause  $X_i \vee X_j \vee X_k$  in  $I_{3SAT}$ . By inverting the roles of  $a_i$  and  $a_{i+n}$  we can clearly impose clauses involving negative literals. This completes the reduction from  $I_{3SAT}$  to the problem of finding a domain ordering of a binary CSP instance so that EMC does not occur. Since this reduction is clearly polynomial, we can conclude that the problem of testing the existence of a domain order so that EMC does not occur is NP-hard.

Now consider the case of the pattern BTX. We use a similar construction to the case of EMC, above. Again, to simulate a clause  $X_i \vee X_j \vee X_k$  in  $I_{3SAT}$  we need to impose



$(a_i > a_{i+n}) \vee (a_j > a_{j+n}) \vee (a_k > a_{k+n})$ . This can be achieved by imposing:

$$\begin{aligned} & (a_i > a_{i+n}) \vee (x_p > x_q), \\ & (a_j > a_{j+n}) \vee (x_q > x_r) \quad \text{and} \\ & (a_k > a_{k+n}) \vee (x_r > x_p) \end{aligned}$$

For example, to impose  $(a_i > a_{i+n}) \vee (x_p > x_q)$  we place the same gadget as above (i.e. positive edges  $(a_i, b)$ ,  $(a_i, c)$ ,  $(a_{i+n}, b)$ ) on each of the three pairs of variables  $(x_p, x_q)$ ,  $(x_{p+1}, x_q)$  and  $(x_{q+1}, x_p)$ . To avoid BTX on variables  $x_p, x_q, x_{q+1}$ , on variables  $x_{p+1}, x_q, x_{q+1}$  and on variables  $x_{q+1}, x_p, x_{p+1}$ , we must have

$$\begin{aligned} & (a_i > a_{i+n}) \vee (x_p > x_q) \vee (x_p > x_{q+1}), \\ & (a_i > a_{i+n}) \vee (x_{p+1} > x_q) \vee (x_{p+1} > x_{q+1}) \quad \text{and} \\ & (a_i > a_{i+n}) \vee (x_{q+1} > x_p) \vee (x_{q+1} > x_{p+1}) \end{aligned}$$

Since there is a total strict ordering on the variables, this is logically equivalent to imposing  $(a_i > a_{i+n}) \vee (x_p > x_q)$ , as required (provided none of the variables  $x_p, x_{p+1}, x_q, x_{q+1}$  are used in other gadgets).

Finally, we consider the pattern BTI. But this is an easier case than BTX. We just need to place the gadget on  $(x_p, x_q)$  to impose  $(a_i > a_{i+n}) \vee (x_p > x_q)$  to avoid the pattern BTI.

Thus, EMC is NP-hard to detect when the domain order of the instance is not fixed, and BTX, BTI are NP-hard to detect when neither the domain order nor the variable order of the instance is fixed.  $\square$

The results from this section are summarised in Table 1. We use the star to denote uninteresting cases. Note that since LX is an unordered pattern the questions of determining variable and/or domain orders are not interesting. Similarly, since pattern BTP only orders variables the question of determining a domain order is not interesting.

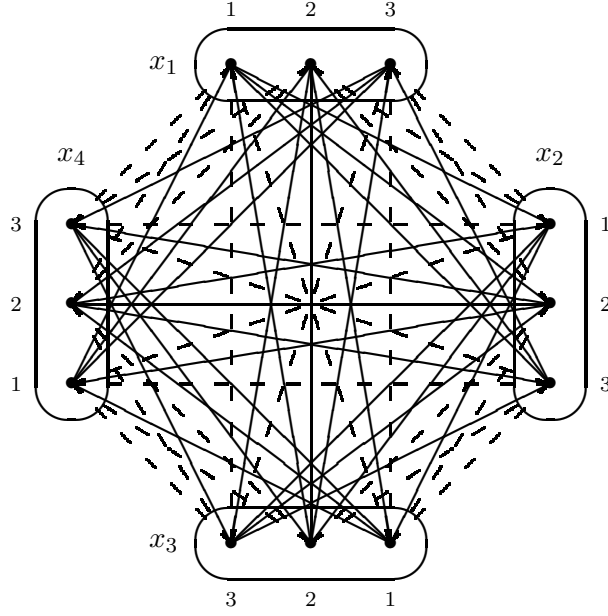
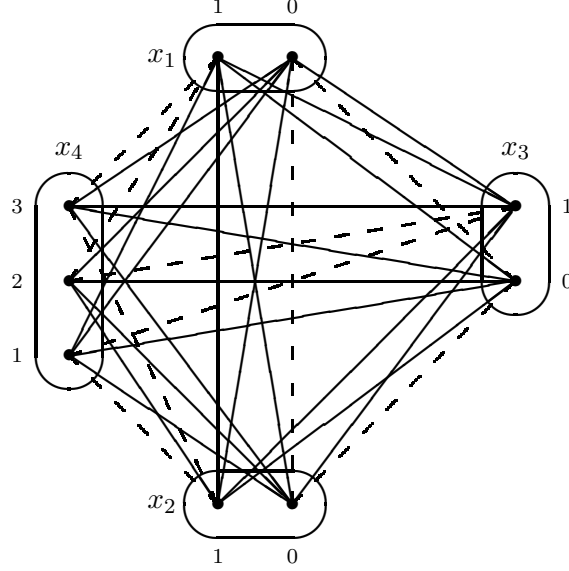
	BTP	BTI	BTX	EMC	LX
domain order given	P [Thm 4.1]	P [Thm 4.2]	P [Thm 4.2]	P [Thm 4.2]	*
variable order given	*	P [Thm 4.3]	P [Thm 4.3]	NP-h [Thm 4.4]	*
no order given	P [Thm 4.1]	NP-h [Thm 4.4]	NP-h [Thm 4.4]	NP-h [Thm 4.4]	*

Table 1: Summary of recognition problems.

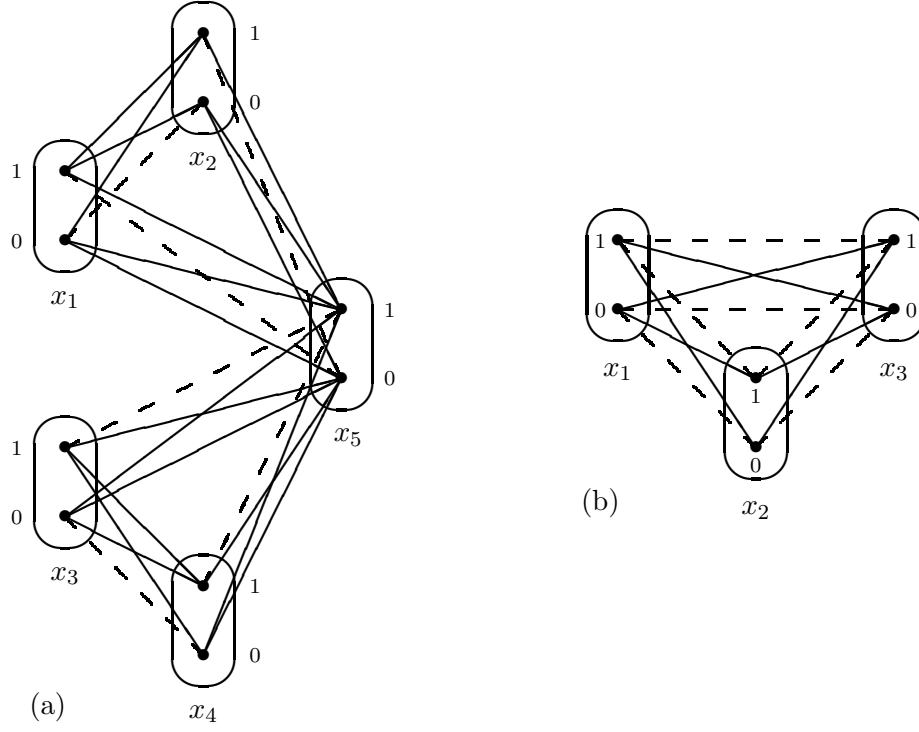
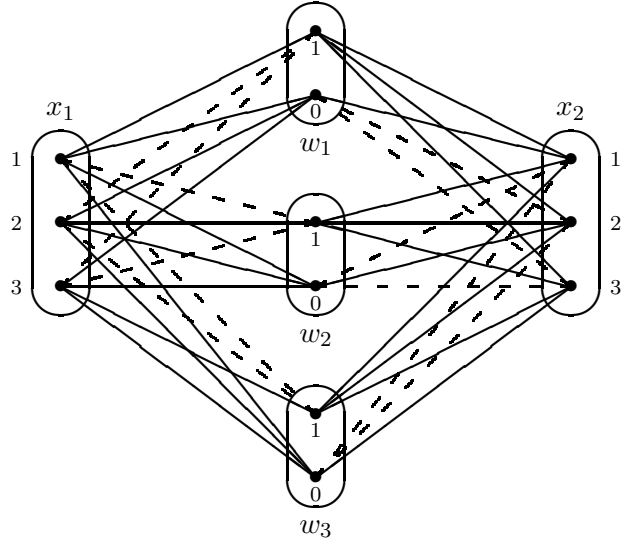
## 5. CHARACTERISATION OF PATTERNS SOLVED BY AC

**5.1. Instances not solved by arc consistency.** We first give a set of instances, each of which is arc consistent and has no solution. If for any of these instances  $I$ , we have  $I \in \text{CSP}_{\overline{SP}}(P)$ , then this constitutes a proof, by Lemma 2.5, that pattern  $P$  is not solved by arc consistency. For simplicity of presentation, in each of the following instances, we suppose the variable order given by  $x_i < x_j$  if  $i < j$ .

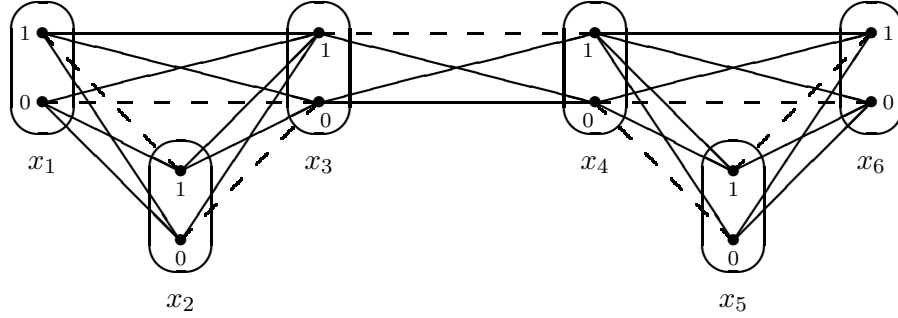
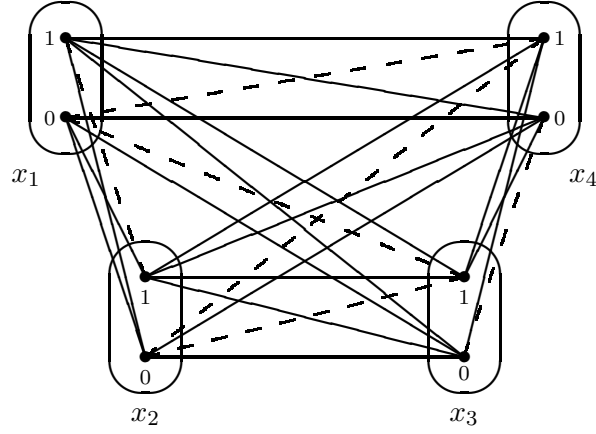
- $I_{K4}$  (shown in Figure 12) is composed of four variables with domains  $D(x_i) = \{1, 2, 3\}$  ( $i = 1, 2, 3, 4$ ), and the following constraints:  $(x_i = 1) \vee (x_j = 3)$   $((i, j) = (1, 2), (2, 3), (3, 4), (4, 1))$  and  $(x_i = 2) \vee (x_j = 2)$   $((i, j) = (1, 3), (2, 4))$ .

Figure 12: The instance  $I_{K4}$ .Figure 13: The instance  $I_4$ .

- $I_4$  (shown in Figure 13) is composed of four variables with domains  $D(x_0) = \{1, 2, 3\}$ ,  $D(x_i) = \{0, 1\}$  ( $i = 1, 2, 3$ ), and the following constraints:  $x_i \vee x_j$  ( $1 \leq i < j \leq 3$ ) and  $(x_0 = i) \vee \overline{x_i}$  ( $i = 1, 2, 3$ ).
- $I_{2\Delta}^{SAT}$  (shown in Figure 14(a)) is composed of five Boolean variables and the following constraints:  $x_1 \vee x_2$ ,  $x_3 \vee x_4$ ,  $\overline{x_1} \vee x_5$ ,  $\overline{x_2} \vee x_5$ ,  $\overline{x_3} \vee \overline{x_5}$ ,  $\overline{x_4} \vee \overline{x_5}$ .

Figure 14: The instances (a)  $I_{2\Delta}^{SAT}$  and (b)  $I_3^{COL}$ .Figure 15: The instance  $I_5$  (with variable order  $w_1 < w_2 < x_1 < x_2 < x_3$ ).

- $I_5$  (shown in Figure 15) is composed of five variables with domains  $D(w_i) = \{0, 1\}$  ( $i = 1, 2, 3$ ),  $D(x_i) = \{1, 2, 3\}$ , and the constraints:  $\overline{w_i} \vee (x_1 = i)$  ( $i = 1, 2, 3$ ) and

Figure 16: The instance  $I_6^{SAT}$ .Figure 17: The instance  $I_{K4}^{SAT}$ .

$w_i \vee (x_2 = i)$  ( $i = 1, 2, 3$ ). In this instance the variable order is  $w_1 < w_2 < x_1 < x_2 < x_3$ .

- $I_6^{SAT}$  (shown in Figure 16) is composed of six Boolean variables and the following constraints:  $\overline{x_1} \vee \overline{x_2}$ ,  $x_1 \vee x_3$ ,  $x_2 \vee x_3$ ,  $\overline{x_3} \vee \overline{x_4}$ ,  $x_4 \vee x_5$ ,  $x_4 \vee x_6$ ,  $\overline{x_5} \vee \overline{x_6}$ .
- $I_{K4}^{SAT}$  (shown in Figure 17) is composed of four Boolean variables and the following constraints:  $\overline{x_1} \vee \overline{x_2}$ ,  $x_3 \vee x_4$  and  $x_i \vee \overline{x_j}$  (for  $(i, j) = (1, 3), (1, 4), (2, 3), (2, 4)$ ).
- $I_3^{COL}$  (shown in Figure 14(b)) is composed of three Boolean variables and the three inequality constraints:  $x_i \neq x_j$  ( $1 \leq i < j \leq 3$ ).

We illustrate four of these instances in Figure 12, Figure 13, Figure 15 and Figure 17. In figures representing CSP instances, similarly to patterns, ovals represent variables, the set of points inside an oval the elements of the variable's domain, a dashed (respectively, solid) line joining two points represents the incompatibility (respectively, compatibility) of the two points. In order not to clutter up figures representing instances, trivial constraints containing only positive edges (i.e. corresponding to complete relations) are not shown. Figure 18(a) is a pattern which does not occur in the instance  $I_{K4}$  (Figure 12). Similarly, Figure 18(b) is a pattern which does not occur in the instance  $I_4$  (Figure 13), and the pattern in Figure 18(c) does not occur in instance  $I_{2\Delta}^{SAT}$ . Figure 18(d), (e) and (f) are

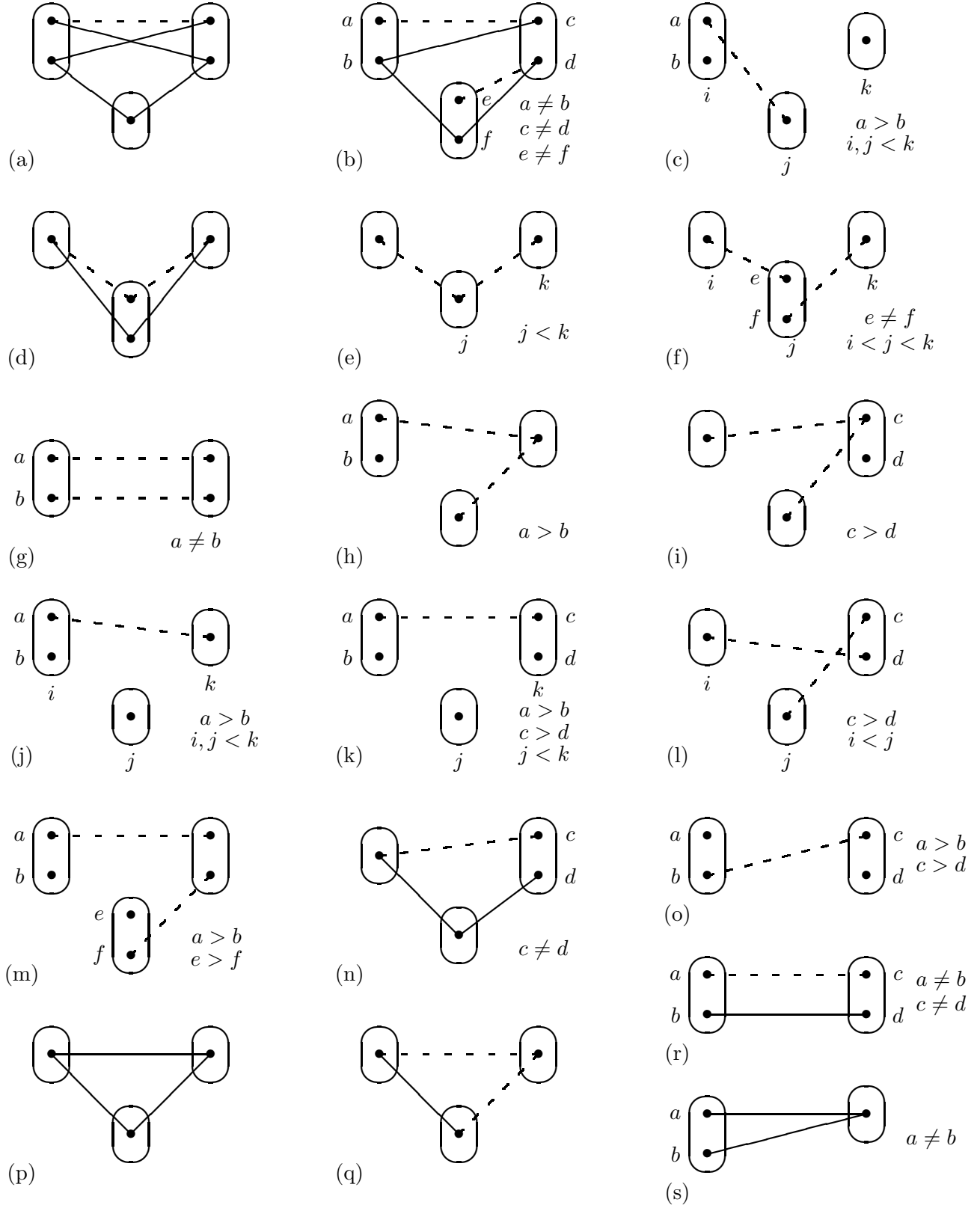


Figure 18: Patterns which do not occur in (a)  $I_{K_4}$ ; (b)  $I_4$ ; (c)  $I_{2\Delta}^{SAT}$ ; (d),(e),(f)  $I_5$ ; (g),(h),(i)  $I_6^{SAT}$ ; (j),(k),(l),(m)  $I_{K_4}^{SAT}$ ; (n),(o),(p),(q),(r),(s)  $I_3^{2COL}$ .

three patterns which do not occur in the instance  $I_5$  (Figure 15). The pattern (known as  $T1$ ) shown in Figure 18(d) is, in fact, a tractable pattern [Cooper and Escamocher(2015)], but the fact that it does not occur in  $I_5$  (an arc-consistent instance which has no solution) shows that arc consistency is not a decision procedure for  $\text{CSP}_{\overline{SP}}(T1)$ . This instance was constructed using certain known properties of the pattern  $T1$  [Escamocher(2014)].

It can easily be verified that the three patterns Figure 18(g), (h), (i) do not occur in  $I_6^{SAT}$ . Similarly, the four patterns in Figure 18(j),(k),(l),(m) do not occur in the instance  $I_{K4}^{SAT}$  (Figure 17).

The instance  $I_3^{2COL}$  is the problem of colouring a complete graph on three vertices with only two colours. It is arc consistent but clearly has no solution. It is easy to verify that none of the six patterns in Figure 18(n),(o),(p),(q),(r),(s) occur in  $I_3^{2COL}$ . Furthermore, trivially, no pattern on four or more variables occurs in  $I_3^{2COL}$  and no pattern with three or more distinct values in the same domain occurs in  $I_3^{2COL}$ .

By Lemma 2.5, we know that if a pattern  $P$  does not occur in any of the instances  $I_{K4}$ ,  $I_4$ ,  $I_{2\Delta}^{SAT}$ ,  $I_5$ ,  $I_6^{SAT}$ ,  $I_{K4}^{SAT}$ ,  $I_3^{2COL}$ , then it is not AC-solvable. Let  $P$  be any of the patterns shown in Figure 18. By Lemma 2.6, any pattern  $Q$  in which  $P$  occurs is not AC-solvable.

By the pattern in Figure 18(g), a simple AC-solvable pattern cannot contain two negative edges between the same pair of variables. Since instance  $I_3^{2COL}$  contains only three variables and instance  $I_5$  contains no triple of variables which have a negative edge between each pair of variables, an AC-solvable pattern can contain *at most three variables and at most two negative edges*. Thus to identify simple AC-solvable patterns we only need to consider patterns on at most three variables, at most two points per variable and with none, one or two negative edges. Furthermore, in the case of two negative edges these negative edges cannot be between the same pair of variables.

**5.2. Characterising AC-solvable unordered patterns.** In this subsection, we consider only patterns  $P$  that have no associated structure (i.e. with  $<_X = <_D = \emptyset$ ). We prove the following characterisation of unstructured AC-solvable patterns.

**Theorem 5.1.** *If  $P$  is a simple unordered pattern, then  $P$  is AC-solvable if and only if  $P$  occurs in the pattern  $LX$  (Figure 11) or in the pattern  $unordered(BTP)$ .*

*Proof.* By the discussion in Section 5.1, we only need to consider patterns  $P$  with at most three variable, at most two points per variable and at most two negative edges (with these edges not being between the same pair of variables). We consider separately the cases of a pattern with 0, 1 or 2 negative edges.

The only simple unordered pattern with no negative edges is the triangle of positive edges shown in Figure 18(p) and this pattern is not AC-solvable since it does not occur in  $I_3^{2COL}$ .

Let  $P$  be a simple pattern with one negative edge  $(a, c)$  (between variables  $y$  and  $z$ ) and at most two points per variable. If the domain of  $y$  (respectively,  $z$ ) contains another point  $b$  (respectively,  $d$ ), then for  $a, b$  (respectively,  $c, d$ ) to be non-mergeable, there must be a positive edge  $(b, c)$  (respectively,  $(a, d)$ ). Furthermore, for  $b$  (respectively,  $d$ ) not to be a dangling point, it must belong to another positive edge. Any two distinct points in the domain of a third variable  $x$  would be mergeable, so we can assume that  $P$  has at most one point in the domain of  $x$ . Since this point is not a dangling point, it must be connected by positive edges to at least two points. By a simple exhaustive search we can easily deduce that *either*  $P$  is (a subpattern of) a triangle on three variables composed of one negative

edge and two positive edges (in which case  $P$  occurs in the pattern LX shown in Figure 11), or one of the patterns shown in Figure 18(a), Figure 18(p) or Figure 18(s) occurs in  $P$ , in which case, by Lemma 2.6,  $P$  is not AC-solvable.

Let  $P$  be a simple pattern containing exactly two negative edges  $(a, b)$  (between variables  $x, y$ ) and  $(a, c)$  (between variables  $x, z$ ) that meet at the point  $a$  of variable  $x$ . Suppose first that  $x$  has no other point. If  $P$  does not occur in LX, then  $P$  must have a positive edge between variables  $y$  and  $z$  which is either  $(b, c)$  or  $(d, e)$  where  $d \neq b$  and  $e \neq c$ . In the latter case, to avoid points  $b, d$  (respectively,  $c, e$ ) being mergeable,  $P$  must have the positive edge  $(a, d)$  (respectively,  $(a, e)$ ). We can deduce that, if  $P$  does not occur in LX, then one of the patterns Figure 18(p) or Figure 18(q) occurs in  $P$ . Suppose now that  $P$  has two points  $a, f$  in the domain of variable  $x$ . For  $a, f$  not to be mergeable,  $P$  must have either the positive edge  $(b, f)$  or the positive edge  $(c, f)$ . If it has both, then the pattern Figure 18(d) occurs in  $P$ . If  $P$  has just one, which without loss of generality we can suppose is the positive edge  $(b, f)$ , then for  $f$  not to be a dangling point,  $f$  must belong to another positive edge  $(d, f)$  (where  $d \neq b$ ) or  $(e, f)$  (where  $e \neq c$ ). In the latter case, for  $c, f$  not to be mergeable,  $P$  must also have the positive edge  $(a, e)$ . In both cases, the pattern Figure 18(s) occurs in  $P$ . Thus, if  $P$  does not occur in LX, then, by Lemma 2.6,  $P$  is not AC-solvable.

Finally, let  $P$  be a simple pattern containing exactly two negative edges  $(a, b)$  (between variables  $x, y$ ) and  $(c, d)$  (between variables  $x, z$ ) with two distinct points  $a \neq c$  in the domain of variable  $x$ . To avoid  $a, c$  being mergeable,  $P$  must have a positive edge  $(b, c)$  or  $(a, d)$ . Without loss of generality, suppose  $P$  has the positive edge  $(b, c)$ . If  $P$  does not occur in unordered(BTP), then at least one of the variables  $y, z$  must have two distinct points. If  $y$  has a point  $e \neq b$ , then for  $b, e$  not to be mergeable,  $P$  must have the positive edge  $(a, e)$ . Similarly, if  $z$  has a point  $f \neq d$ , then  $P$  must have the positive edge  $(c, f)$ . But then, to avoid dangling points, we have to add other positive edges to  $P$  and we find that one of the patterns Figure 18(a), Figure 18(n), Figure 18(p) or Figure 18(s) occurs in  $P$ , and so, by Lemma 2.6,  $P$  is not AC-solvable. By Lemma 2.4, if  $P$  occurs in unordered(BTP), then  $P$  occurs in BTP, and thus is AC-solvable [Cooper et al.(2010b)].  $\square$

**5.3. Characterising AC-solvable variable-ordered patterns.** In this subsection we consider simple patterns  $P$  which have no domain order, (i.e.  $<_D = \emptyset$ ), but do have a partial order on the variables. We first require the following lemma.

**Lemma 5.2.** *If  $P^<$  is a pattern whose only structure is a partial order on its variables and  $P^- = \text{unordered}(P^<)$ , then*

- (1)  $P^<$  is simple if and only if  $P^-$  is simple.
- (2)  $P^<$  is AC-solvable only if  $P^-$  is AC-solvable.

*Proof.* The property of being simple is (Definition 9) independent of any variable order, hence  $P^<$  is simple if and only if  $P^-$  is simple. By Lemma 2.4,  $P^-$  occurs in  $P^<$ . The fact that  $P^<$  is AC-solvable only if  $P^-$  is AC-solvable then follows from Lemma 2.6.  $\square$

Recall pattern LX<sup><</sup> from Example 7 that is obtained from the pattern LX (Figure 11) by adding the partial variable order  $y < z$ .

Lemma 5.2 allows us to give the following characterisation of variable-ordered AC-solvable patterns.

**Theorem 5.3.** *If  $P$  is a simple pattern whose only structure is a partial order on its variables, then  $P$  is AC-solvable if and only if  $P$  occurs in the pattern  $LX^<$  (Example 7), the pattern  $BTP^{vo}$  (Figure 2) or the pattern  $\text{invVar}(BTP^{vo})$ .*

*Proof.* By Lemma 5.2 and Theorem 5.1, we only need to consider patterns  $P^-$  occurring in  $LX$  or  $\text{unordered}(BTP)$  to which we add a partial order on the variables to produce a pattern  $P$ .

We first consider the case of a pattern  $P$  in which two negative edges that meet. By Lemma 5.2 and Theorem 5.1, for  $P$  to be AC-solvable,  $P^-$  must occur in  $LX$ . By Theorem 3.4 and Example 7, we have that  $LX^<$  is AC-solvable. Let  $L$  be the pattern composed of three variables  $x, y, z$  and two negative edges which meet at a point  $\langle x, \epsilon \rangle$  (i.e.  $L$  is the pattern  $LX$  without its positive edges). Then Figure 18(e) and Lemma 2.7 tell us that placing any order between  $x$  and  $y$  or between  $x$  and  $z$  turns  $L$  into a pattern which is not AC-solvable. It follows from Lemma 2.6 that an ordered pattern  $P$  containing two negative edges that meet is AC-solvable if and only if  $P$  occurs in  $LX^<$ .

Now consider simple patterns  $P$  which contain two negative edges that do not meet. By Lemma 5.2 and Theorem 5.1, for  $P$  to be AC-solvable,  $P^-$  must occur in  $\text{unordered}(BTP)$ . Adding almost any partial variable order to  $\text{unordered}(BTP)$  produces a pattern which occurs in  $BTP^{vo}$  (Figure 2(b)) or  $\text{invVar}(BTP^{vo})$ . The only order for which this is not the case, is the total order  $x < z < y$  (or its inverse), where the variables  $x, y, z$  are as shown in Figure 2(b). Let  $P$  be any simple pattern on three variables  $x, y, z$ , containing two negative edges (between  $x, z$  and  $y, z$ ) that do not meet and with the variable order  $x < z < y$ . Then the pattern shown in Figure 18(f) occurs in  $P$  and hence, by Lemma 2.6,  $P$  is not AC-solvable.

If  $P$  is any simple pattern such that  $P^-$  occurs in  $LX$  or  $\text{unordered}(BTP)$  and contains at most one negative edge, then  $P$  occurs in the three-variable triangle pattern composed of one negative and two positive edges; it is then easy to check that, whatever the ordering of its variables,  $P$  occurs in  $BTP^{vo}$  or  $\text{invVar}(BTP^{vo})$ .  $\square$

**5.4. Characterising AC-solvable domain-ordered patterns.** In this subsection we consider simple patterns  $P$  with a partial order on domains but no ordering on the variables.

Let  $\text{EMC}^-$  be the no-variable-order version of the pattern  $\text{EMC}$  depicted in Figure 3. We prove the following characterisation of domain-ordered AC-solvable patterns.

**Theorem 5.4.** *If  $P$  is a simple pattern whose only structure is a partial order on its domains, then  $P$  is AC-solvable if and only if  $P$  occurs in the pattern  $LX$  (Figure 11), or the pattern  $\text{EMC}^-$ , or the pattern  $\text{invDom}(\text{EMC}^-)$ .*

*Proof.* As in the proofs of Theorem 5.1 and 5.3, we only need to consider patterns on at most three variables, with at most two points per variable and with either no negative edges, one negative edge, two negative edges that meet or two negative edges that do not meet. However, we have more cases to consider than in Theorem 5.1 since patterns may now contain points, such as  $\beta$  in the pattern  $\text{EMC}$  shown in Figure 3, which would be a dangling point without the domain order  $\alpha > \beta$ .

The only pattern with no negative edges and no mergeable points is the triangle of positive edges shown in Figure 18(p) which is not AC-solvable.



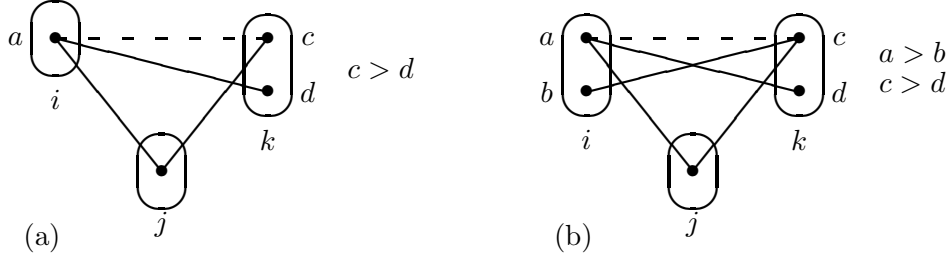
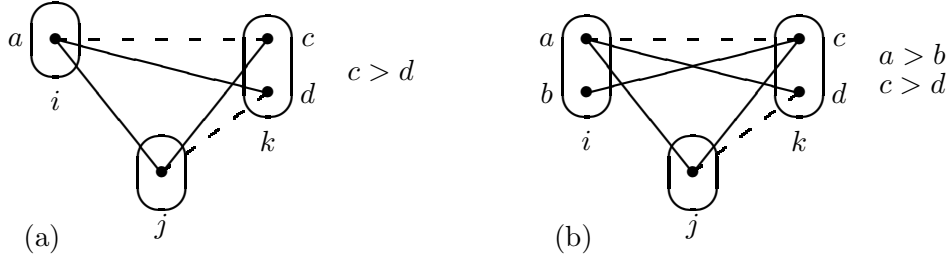
Let  $P$  be a simple pattern with one negative edge, at most two points per variable and just two variables. If neither of the patterns shown in Figure 18(o) and Figure 18(r) occur in  $P$ , then  $P$  occurs in the pattern  $\text{EMC}^-$  or in  $\text{invDom}(\text{EMC}^-)$ .

Let  $P$  be a simple pattern on three variables  $x, y, z$ , with one negative edge  $(a, c)$  (between variables  $y$  and  $z$ ) and at most two points per variable. If the domain of  $y$  (respectively,  $z$ ) contains another point  $b$  (respectively,  $d$ ), then for  $a, b$  (respectively,  $c, d$ ) to be non-mergeable, there must be a positive edge  $(b, c)$  (respectively,  $(a, d)$ ). Any two distinct points in the domain of variable  $x$  would be mergeable, so we can assume that  $P$  has exactly one point in the domain of  $x$ . Since this point  $e$  is not a dangling point, it must be connected by positive edges to at least two points. If none of the patterns in Figure 18(a), Figure 18(p) and Figure 18(s) occurs in  $P$ , then  $e$  must belong to the two positive edges  $(a, e)$  and  $(c, e)$ , and no others. If neither of the patterns in Figure 18(o) or Figure 18(r) occurs in  $P$ , then we can deduce that  $P$  occurs in the pattern  $\text{EMC}^-$  or in  $\text{invDom}(\text{EMC}^-)$ .

Let  $P$  be a simple pattern on three variables with at most two points per variable and with two negative edges that meet. If  $P$  has two points  $a, b$  in the domain of the same variable together with an ordering  $a < b$ , then one of the patterns in Figure 18(h) and Figure 18(i) (or their domain-inversed version) occurs in  $P$ , and hence  $P$  cannot be AC-solvable. This leaves only the case of unordered patterns  $P$ . By the proof (and in particular the part that deals with patterns containing exactly two negative edges that meet) of Theorem 5.1, we can deduce that if  $P$  is AC-solvable then it occurs in the pattern LX.

Let  $P$  be a simple pattern on three variables  $x, y, z$  with at most two points per variable and with two negative edges  $(a, c)$  (between variables  $x$  and  $y$ ) and  $(d, f)$  (between variables  $y$  and  $z$ ) that do not meet (i.e.  $c \neq d$ ). If  $P$  contains only these four points  $a, c, d, f$ , then it necessarily occurs in  $\text{EMC}^-$ . If  $P$  contains exactly five points, then without loss of generality, we can assume that there is a point  $b \neq a$  in the domain of  $x$ . Since  $a, b$  are not mergeable, there must be a positive edge  $(b, c)$  in  $P$ . If  $P$  has a positive edge  $(b, f)$ , then the pattern in Figure 18(n) occurs in  $P$ ; if  $P$  has a positive edge  $(b, d)$  then the pattern in Figure 18(s) occurs in  $P$ . Now, if the pattern in Figure 18(o) does not occur in  $P$ , then whatever ordering is placed on  $a, b$  and  $c, d$ ,  $P$  occurs in the pattern  $\text{EMC}^-$  or in  $\text{invDom}(\text{EMC}^-)$ . If  $P$  contains exactly six points, then there must be points  $b \neq a$  in the domain of  $x$  and  $e \neq f$  in the domain of  $z$ . Since both  $a, b$  and  $e, f$ , are not mergeable, there must be positive edges  $(b, c)$  and  $(e, d)$ . If  $P$  has a positive edge  $(b, e)$ , then the pattern in Figure 18(b) occurs in  $P$ ; if  $P$  has a positive edge  $(b, f)$  or  $(a, e)$ , then the pattern in Figure 18(n) occurs in  $P$ ; if  $P$  has a positive edge  $(b, d)$  or  $(e, c)$ , then the pattern in Figure 18(s) occurs in  $P$ . But then in all other cases one of  $b$  and  $e$  is a dangling point unless  $P$  has an order on both  $a, b$  and  $e, f$ . But this then implies that at least one of the patterns in Figure 18(m) and Figure 18(o) occur in  $P$ . In all these cases, by Lemma 2.6,  $P$  is not AC-solvable.  $\square$

**5.5. Characterising AC-solvable ordered patterns.** In this subsection we consider the most general case of simple patterns  $P$  which have a partial domain order and a partial variable order. We prove the following characterisation of AC-solvable patterns with partial orders on domains and variables.

Figure 19: One-negative-edge patterns occurring in  $\text{EMC}^-$ .Figure 20: Two-negative-edge patterns occurring in  $\text{EMC}$ .

**Theorem 5.5.** *If  $P$  is a simple pattern with a partial order on its domains and/or variables, then  $P$  is AC-solvable if and only if  $P$  occurs in one of the patterns  $\text{LX}^<$ ,  $\text{EMC}$  (Figure 3),  $\text{BTP}^{\text{vo}}$ ,  $\text{BTP}^{\text{do}}$  (Figure 2),  $\text{BTX}$  (Figure 7) or  $\text{BTI}$  (Figure 8) (or versions of these patterns with inversed domain-order and/or variable-order).*

*Proof.* Let  $P^-$  be the same pattern as  $P$  but without the partial order on its variables. If  $P$  is AC-solvable, then, by Lemma 2.4,  $P^-$  occurs in  $P$ , and hence, by Lemmas 2.3 and 2.5,  $P^-$  is also AC-solvable. Thus, by Theorem 5.4,  $P$  must occur in either  $\text{LX}$ ,  $\text{EMC}^-$  or  $\text{invDom}(\text{EMC}^-)$ . We consider the four cases: no negative edges, one negative edge, two negative edges that meet, two negative edges that do not meet in  $P$ . We have already seen in the proofs of Theorems 5.1 and 5.4 that there are no simple AC-solvable patterns with only positive edges, so there remain three cases to consider.

Let  $P$  be a simple pattern with exactly one negative edge. If  $P^-$  (which also has one negative edge) occurs in  $\text{LX}$  and is simple (and hence has no dangling points) then it must occur in a triangle  $T$  consisting of one negative and two positive edges. This triangle pattern  $T$  occurs in  $\text{BTP}$  or in  $\text{invVar}(\text{BTP})$  whatever ordering we place on its three variables and hence the same is true of  $P$ . If  $P^-$  (which has one negative edge and is unmergeable) occurs in  $\text{EMC}^-$  but not in the triangle  $T$ , then  $P^-$  occurs in one of the two patterns shown in Figure 19 (or the domain-inversed versions of these patterns) and includes the point  $d$  (in the case corresponding to Figure 19(a)) or the points  $b$  and  $d$  (in the case corresponding to Figure 19(b)). For  $d$  (respectively,  $b$ ) not to be a dangling point in  $P$ , we must have the order  $c > d$  (respectively  $a > b$ ) in  $P$ . If  $P^-$  occurs in the pattern in Figure 19(a) and  $P$  has the variable order  $i, j < k$  (or  $i < k$  or  $j < k$ ), then  $P$  occurs in  $\text{BTP}^{\text{do}}$ . If  $P^-$  occurs in the pattern in Figure 19(a) and  $P$  has the variable order  $i < j$ , then  $P$  occurs in  $\text{invVar}(\text{BTI})$ . If  $P^-$  occurs in the pattern in Figure 19(a) and  $P$  includes the variable order  $i < j, k$  and domain order  $c > d$ , then the pattern in Figure 18(j) occurs in  $\text{invVar}(P)$ . If  $P^-$  occurs in

the pattern in Figure 19(a) and  $P$  includes the variable order  $i, k < j$  and domain order  $c > d$ , then the pattern in Figure 18(c) occurs in  $P$ . This covers all variable orderings of  $P$  (after taking into account the variable-inversed versions of each case) when  $P^-$  occurs in the pattern in Figure 19(a). Now consider the case in which  $P^-$  occurs in the pattern in Figure 19(b). If  $P$  includes the variable order  $j < k$  (or  $j < i$ ) together with the domain order  $a > b$  and  $c > d$ , then the pattern in Figure 18(k) occurs in  $P$ . If  $P$  has the variable order  $i < k$  then  $P$  occurs in EMC. Thus all one-negative-edge AC-solvable patterns occur in BTP, BTI or EMC (or their domain and/or variable-inversed versions).

Let  $P$  be a simple pattern with two negative edges that meet at a point.  $P^-$  necessarily occurs in the pattern LX. Let  $j$  be the variable of  $P$  where the two negative edges meet, and let  $i, k$  be the other two variables. If  $P$  includes the variable order  $j < k$  (or, by symmetry, the order  $j < i$ ), then the pattern in Figure 18(e) occurs in  $P$  and hence  $P$  is not AC-solvable. If  $P$  has the variable order  $i < k$ , then  $P$  occurs in the pattern  $LX^<$  (the version of LX shown in Figure 11 together with the variable order  $y < z$ ). By symmetry, we have covered all possible cases.

Finally, let  $P$  be a simple pattern with two negative edges that do not meet at a point.  $P^-$  necessarily occurs in the pattern EMC. We distinguish two distinct cases: (1)  $P$  occurs in the pattern in Figure 20(a) or (2)  $P$  occurs in the pattern in Figure 20(b) and includes the point  $b$  together with the order  $a > b$  (otherwise  $b$  would be a dangling point). We first consider case (1). If  $P$  includes the order  $c > d$  and  $i < j$ , then the (inversed domain-order version of the) pattern in Figure 18(l) occurs in  $P$  and hence  $P$  is not AC-solvable. If  $P$  includes the order  $i < k < j$ , then the pattern in Figure 18(f) occurs in  $P$ . If  $P$  has the variable order  $i < j < k$  and no domain order, then  $P$  occurs in  $BTP^{vo}$ . If  $P$  has the variable order  $i, j < k$  and the domain order  $c > d$ , then  $P$  occurs in  $BTP^{do}$ . All other patterns which fall in case (1) are covered by symmetry. Now we consider the case (2). First suppose that  $P$  includes the domain order  $c > d$  (as well as  $a > b$ ). If  $P$  includes the variable order  $i < j$ , then the (domain-inversed version of the) pattern in Figure 18(l) occurs in  $P$ . If  $P$  includes the variable order  $j < k$ , then the pattern in Figure 18(k) occurs in  $P$ . If  $P$  has the variable order  $i < k$ , then  $P$  occurs in EMC. Now, suppose that  $P$  does not include the domain order  $c > d$ . If  $P$  includes the variable order  $i, j < k$ , then the pattern in Figure 18(j) occurs in  $P$ . If  $P$  includes the variable order  $i, k < j$ , then the pattern in Figure 18(c) occurs in  $P$ . If  $P$  has the variable order  $i < j, k$  (or  $i < j$  or  $i < k$ ) then  $P$  occurs in BTX. If  $P$  has the variable order  $j < k$  then  $P$  occurs in BTI. By symmetry we have covered all possible variable orderings of  $P$  in case (2).  $\square$

## 6. CONCLUSION

We have identified 4 new tractable classes of binary CSPs. Moreover, we have given a characterisation of all simple partially-ordered patterns decided by AC. We finish with open problems.

For future work, we plan to study the wider class of unmergeable ordered patterns in which two points  $a, b$  may be non-mergeable simply because there is an order  $a < b$  on them. In the present paper,  $a, b$  are mergeable unless they have different compatibilities with a third point  $c$ .

Is there a way of combining EMC, BTX and BTI, since to find a solution after establishing arc consistency we use basically the same algorithm? Any such generalisation will

not be a simple forbidden pattern by Theorem 5.5, but there is possibly some other way of combining these patterns.

Are there interesting generalisations of these patterns to constraints of arbitrary arity, valued constraints, infinite domains or QCSP? BTP has been generalised to constraints of arbitrary arity [Cooper et al.(2014)] as well as to QCSPs [Gao et al.(2011)]. Max-closed constraints have been generalised to VCSPs [Cohen et al.(2006)]. Infinite domains is an interesting avenue of future research because simple temporal constraints are binary max-closed [Dechter et al.(1991)].

In this paper, we only focused on classes of CSP instances with totally ordered domains (but defined by partially-ordered patterns). However, the framework of forbidden patterns captures language-based CSPs with partially-ordered domains, such as CSPs with a semi-lattice polymorphism. In the future, we plan to investigate classes of CSP instances with partially-ordered domains.

## REFERENCES

- [Atserias et al.(2007)] A. Atserias, A. A. Bulatov, and V. Dalmau. On the Power of  $k$ -Consistency. In Proc. IICALP'07, 279–290. Springer, 2007.
- [Barto(2014)] L. Barto. Constraint satisfaction problem and universal algebra. *ACM SIGLOG News*, 1(2): 14–24, 2014.
- [Barto and Kozik(2014)] L. Barto and M. Kozik. Constraint Satisfaction Problems Solvable by Local Consistency Methods. *J.ACM*, 61(1), 2014. Article No. 3.
- [Bessière et al.(2005)] C. Bessière, J.-C. Régin, R. H. C. Yap, and Y. Zhang. An optimal coarse-grained arc consistency algorithm. *Artif. Intel.*, 165(2):165–185, 2005.
- [Bulatov(2006)] A. Bulatov. Combinatorial problems raised from 2-semilattices. *Journal of Algebra*, 298: 321–339, 2006.
- [Bulatov(2009)] A. Bulatov. Bounded relational width. Unpublished manuscript, 2009.
- [Bulatov and Dalmau(2006)] A. Bulatov and V. Dalmau. A simple algorithm for Mal'tsev constraints. *SIAM Journal on Computing*, 36(1):16–27, 2006.
- [Carbonnel and Cooper(2015)] C. Carbonnel and M. C. Cooper. Tractability in constraint satisfaction problems: a survey. *Constraints*. To appear.
- [Cohen et al.(2006)] D. A. Cohen, M. C. Cooper, P. G. Jeavons, and A. A. Krokhin. The Complexity of Soft Constraint Satisfaction. *Artificial Intelligence*, 170(11):983–1016, 2006.
- [Cohen et al.(2012)] D. A. Cohen, M. C. Cooper, P. Creed, D. Marx, and A. Z. Salamon. The tractability of CSP classes defined by forbidden patterns. *Journal of Artificial Intelligence Research*, 45:47–78, 2012.
- [Cohen et al.(2015a)] D. A. Cohen, M. C. Cooper, G. Escamocher, and S. Živný. Variable and value elimination in binary constraint satisfaction via forbidden patterns. *Journal of Computer and System Sciences*, 81(7):1127–1143, 2015a.
- [Cohen et al.(2015b)] D. A. Cohen, M. C. Cooper, P. Jeavons, and S. Živný. Tractable classes of binary CSPs defined by excluded topological minors. In Proc. IJCAI'15, 1945–1951. AAAI Press, 2015b.
- [Cooper(1999)] M. C. Cooper. Linear-time algorithms for testing the realisability of line drawings of curved objects. *Artificial Intelligence*, 108:31–67, 1999.
- [Cooper and Escamocher(2015)] M. C. Cooper and G. Escamocher. Characterising the complexity of constraint satisfaction problems defined by 2-constraint forbidden patterns. *Discrete Applied Mathematics*, 184:89–113, 2015.
- [Cooper et al.(2010a)] M. C. Cooper, S. de Givry, M. Sánchez, T. Schiex, M. Zytnicki, and T. Werner. Soft arc consistency revisited. *Artificial Intelligence*, 174(7–8):449–478, 2010a.
- [Cooper et al.(2010b)] M. C. Cooper, P. G. Jeavons, and A. Z. Salamon. Generalizing constraint satisfaction on trees: Hybrid tractability and variable elimination. *Artificial Intelligence*, 174(9–10):570–584, 2010b.
- [Cooper et al.(2014)] M. C. Cooper, A. El Mouelhi, C. Terrioux, and B. Zanuttini. On broken triangles. In Proc. CP'14, 9–24. Springer, 2014.
- [Cooper et al.(2015b)] M. C. Cooper, P. Jégou, and C. Terrioux. A Microstructure-Based Family of Tractable Classes for CSPs. In Proc. CP'15, 74–88. Springer, 2015b.

- [Dalmau(2006)] V. Dalmau. Generalized Majority-Minority Operations are Tractable. *Logical Methods in Computer Science*, 2(4), 2006.
- [Dechter et al.(1991)] R. Dechter, I. Meiri, and J. Pearl. Temporal Constraint Networks. *Artificial Intelligence*, 49:61–95, 1991.
- [Escamocher(2014)] G. Escamocher. *Forbidden Patterns in Constraint Satisfaction Problems*. PhD thesis, University of Toulouse, 2014.
- [Feder and Vardi(1998)] T. Feder and M. Y. Vardi. The Computational Structure of Monotone Monadic SNP and Constraint Satisfaction: A Study through Datalog and Group Theory. *SIAM Journal on Computing*, 28(1):57–104, 1998.
- [Gao et al.(2011)] J. Gao, M. Yin, and J. Zhou. Hybrid tractable classes of binary quantified constraint satisfaction problems. In Proc. AAAI’11. AAAI Press, 2011.
- [Green and Cohen(2008)] M. J. Green and D. A. Cohen. Domain permutation reduction for constraint satisfaction problems. *Artif. Intel.*, 172(8-9):1094–1118, 2008.
- [Grohe(2007)] M. Grohe. The complexity of homomorphism and constraint satisfaction problems seen from the other side. *J.ACM*, 54(1):1–24, 2007.
- [Idziak et al.(2010)] P. M. Idziak, P. Markovic, R. McKenzie, M. Valeriote, and R. Willard. Tractability and learnability arising from algebras with few subpowers. *SIAM Journal on Computing*, 39(7):3023–3037, 2010.
- [Jeavons and Cooper(1995)] P. G. Jeavons and M. C. Cooper. Tractable Constraints on Ordered Domains. *Artificial Intelligence*, 79(2):327–339, 1995.
- [Kolmogorov et al.(2015)] V. Kolmogorov, M. Rolínek, and R. Takhanov. Effectiveness of structural restrictions for hybrid CSPs. *CoRR*, abs/1504.07067, 2015.
- [Kolmogorov et al.(2015b)] V. Kolmogorov, J. Thapper, and S. Živný. The power of linear programming for general-valued CSPs, *SIAM Journal on Computing*, 44(1):1–36, 2015.
- [Marx(2013)] D. Marx. Tractable hypergraph properties for constraint satisfaction and conjunctive queries. *Journal of the ACM*, 60(6), 2013. Article No. 42.
- [Naanaa(2013)] W. Naanaa. Unifying and extending hybrid tractable classes of CSPs. *J. of Experimental and Theoretical Artif. Intel.*, 25(4):407–424, 2013.
- [Rossi et al.(2006)] F. Rossi, P. van Beek, and T. Walsh, editors. *The Handbook of Constraint Programming*. Elsevier, 2006.
- [Takhanov(2015)] R. Takhanov. Hybrid (V)CSPs and algebraic reductions. *CoRR*, abs/1506.06540, 2015.
- [Thapper and Živný(2016)] J. Thapper and S. Živný. The complexity of finite-valued CSPs. *J.ACM*. To appear.